1

MODELING THE WAKE AS A CONTINUOUS VORTEX SHEET IN A POTENTIAL-FLOW SOLUTION USING VORTEX PANELS

THESIS

Robert J. Papka
Major, USAF

AFIT/GAE/ENY/89D-26

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

1

AFIT/GAE/ENY/89D-26

MODELING THE WAKE AS A CONTINUOUS VORTEX SHEET IN A POTENTIAL-FLOW SOLUTION USING VORTEX PANELS

THESIS

Robert J. Papka
Major, USAF

AFIT/GAE/ENY/89D-26

AFIT/GAE/ENY/89D-26

MODELING THE WAKE AS A CONTINUOUS VORTEX SHEET IN A POTENTIAL-FLOW SOLUTION USING VORTEX PANELS

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Aeronautical Engineering

Robert J. Papka, B.S.
Major, USAF

December 1989

Approved for public release; distribution unlimited

Acknowledgements

I wish to express my sincere appreciation to many who made this work possible:

To Capt Curt Mracek for his advice, guidance, and indefatigable patience with me through this work, especially in light of his own difficult times. He taught me lessons far more valuable than the academic.

To my committee members, Lt Col. DeJongh and Capt. Phil Beran for their hours, wisdom, and candidness in editing the manuscript.

To my classmates who invested so much of their valuable time to help me through.

Especially to my family -- for the constant prayers and encouragement of my mother, brother, and sister. To Bill and Annette Harrison for their prayer and counsel. To my children, Brad, Jason, and Amber -- I joyfully anticipate being their full-time father again. Most of all, to my wife, Cathy, whose untiring love provided a shoulder to cry on and chastisement when I procrastinated, and who always rejoiced with me at the victories.

No humor intended, my completing this thesis and AFIT are nothing short of a miracle. Without the grace of God, I would not have made it. To my Lord and Savior, Jesus Christ, be all the glory for whatever I may accomplish.

Table of Contents

List of Figures

Abstract

An investigation was made to determine the advantages of modeling the wake as a continuous vortex sheet in an unsteady potential-flow computer model. The existing program modeled the body as a continuous vortex sheet, and modeled the wake as discrete vortex cores employing principles of vortex-lattice methods. A method is developed to approximate the wake with triangular vortex panels. The method accounts for redistributing vortex strength as the panels expand and deform. Comparisons are made with results of the original program for rectangular wings at varying angles of attack. The simulation was attempted for delta wings, but encountered numerical difficulties. The limited results suggest that the theory and method are valid, and recommended areas are given for further pursuit.

MODELING THE WAKE AS A CONTINUOUS VORTEX SHEET IN A POTENTIAL-FLOW SOLUTION USING VORTEX PANELS

I. Introduction

As the cost of producing and flight-testing prototypes has become more and more prohibitive, the increasing power and affordability of computers make them an invaluable alternative in the engineering design process. To date, a great deal of experimental data is available and is used to predict the aerodynamic behavior of a design, but a robust mathematical model, particularly in the realm of vortex-dominated flows, would be a very useful tool. The present work attempts improvement of an existing method developed by Curtis Mracek (1) in 1988. He developed an highly successful unsteady, potential-flow aerodynamic model based on vortex panel methods. This Fortran program models the wake development by way of discrete vortex cores after flow is started impulsively, and accurately predicts the pressure distribution in vortex-dominated flow over wings. The present research pursues Mracek's recommendation to model the wake as a continuous vortex sheet.

Motivation for Research

In addition to the financial benefits of computer simulation already mentioned, there are definite time and convenience returns to be exploited in the design process if an accurate computer model exists. Mracek's model was a marked advance in that it extended previous methods limited to steady and quasi-steady flows to general unsteady flow, and introduced control surface motion and aerodynamic-dynamic simulation. Vortex-dominated flows are of particular interest as high angle of attack capabilities are sought in modern aircraft. As Mracek states,

> Presently there are several good methods for predicting steady aerodynamic loads on aircraft at low angles of attack for slow speeds...The motivation for investigating higher angles of attack with vortex dominated flow is that many modern aircraft routinely fly in this regime, and recently efforts have been made to harness the vorticity-induced forces by the use of strakes, leading edge blowing, and vortex channels. Most numerical methods have trouble predicting the forces and moments for these conditions, and most cannot be used for unsteady motion. One reason they have trouble at high angles of attack is the flow around the wing is heavily influenced by the vortex separation along the leading edge. (1:2)

The existing program is an hybrid vortex method in that the body (wing) is discretized into triangular panels of vorticity that varies linearly, and the wake is modeled as a mesh of discrete vortex cores using a generalization of the procedure employed in the unsteady vortex-lattice method. will be shown later, the vortex panel method has two major advantages over using discrete vortex cores that this research attempts to exploit:

(1) The disturbance velocity induced by a vortex panel approaches infinity at a slower rate as the distance from the panel decreases to zero since it is a logrithmic singularity. Hence, artificial "cutoff" distances inside which induced velocities are arbitrarily set to zero can be reduced.

(2) The disturbance velocities induced by a vortex panel are more smoothly distributed than those of discrete cores very near the vorticity itself. Hence, nature is more closely approximated.

Scope of Development

Even though the previously existing program incorporates the dynamics of moving control surfaces and roll-, pitch-, and yaw-rates, this development considers only impulsively started uniform flow. The focus is on building a continuous vortex sheet to model the wake similar to the bound vorticity approximation of the body. This is done by using the vorticities already calculated by the program (and convected into the wake as discrete cores). Using bound vortex panels to calculate induced velocities is much more expensive in terms of computer time and memory than using discrete vortex cores. Therefore, the method developed is only to be used for points of interest near the vorticity itself where the above advantages are realized. This necessitates a study to determine "how close is too close" for the discrete core method.

II. Aerodynamic Mathematical Model

It is neither the intent nor within the scope of this report to provide an exhaustive development of vortex-panel or vortex-lattice methods. A brief summary follows of the basic principles employed by the existing program.

Governing Equations

The program is a potential-flow solution using vortex distributions. Using the development of Karamcheti (2:254), the following field equations define the flow. The continuity equation for incompressible flow is

$$\text{div } \vec{V} = 0 \qquad (1)$$

where $\vec{V}$ is the total velocity of the fluid. The total velocity at any point in the flow is the sum of the freestream velocity and the disturbance velocities induced by any bodies immersed in the flow.

$$\begin{aligned}\vec{V} &= \vec{V}_{disturbance} + \vec{V}_{freestream} \\ &= \vec{V}_d + \vec{V}_{fs} \qquad (2)\end{aligned}$$

The vorticity field, $\vec{\Omega}$, is defined as

$$\vec{\Omega} = \text{curl } \vec{V} = \text{curl}(\vec{V}_d + \vec{V}_{fs}) \qquad (3)$$

Since the freestream is irrotational, Equation 3 reduces to

$$\vec{\Omega} = \text{curl } \vec{V}_d \tag{4}$$

A third vector field, $\vec{A}$, is chosen such that

$$\vec{V} = \text{curl } \vec{A} \tag{5}$$

The continuity equation then becomes

$$\text{div}(\text{curl } \vec{A}) = 0 \tag{6}$$

which is satisfied for any $\vec{A}$. Now the problem of finding the disturbance velocity reduces to finding the solution to

$$\vec{\Omega} = -\nabla^2\vec{A} \tag{7}$$

As long as div $\vec{\Omega} = 0$, Green's Theorem gives

$$\vec{A}(\vec{r},t) = \frac{1}{4\pi} \iiint_R \frac{\vec{\Omega}(\vec{r},t)}{|\vec{r}-\vec{s}|} \, d\tau \tag{8}$$

where $\vec{\Omega}$ is the vcrticity at $\vec{s}$, $\vec{r}$ is the point of interest, and R is the region containing the vorticity. Karamcheti (2:530-532) establishes that if the vorticity, $\vec{\Omega}$, is contained in a region of thickness, ϵ, adjacent to the body surface, in the limiting process as ϵ approaches zero, the product of $|\vec{\Omega}|$ and ϵ remains constant and $\vec{\Omega}$ becomes tangent to the surface. Define

$$\gamma(\vec{s}) = \lim_{|\vec{\Omega}| \to \infty,\, \epsilon \to 0} |\vec{\Omega}|\epsilon \tag{9}$$

Then Equation (8) can be rewritten as a surface integral

$$\vec{A}(\vec{r}) = \frac{1}{4\pi} \iint_S \frac{\vec{\gamma}(\vec{s})}{|\vec{r}-\vec{s}|} \, d\sigma \tag{10}$$

where S is the surface of the body. The disturbance velocity induced by a continuous vortex sheet, then, is

$$V_d(\vec{r}) = \text{curl } \vec{A} = \frac{1}{4\pi} \text{ curl} \iint_S \frac{\vec{\gamma}(\vec{s})}{|\vec{r}-\vec{s}|} \, d\sigma \tag{11}$$

Since the divergence of the curl of a vector is identically zero, any vector field that represents the surface vorticity will satisfy the continuity equation as long as

$$\text{div } \vec{\gamma} = 0 \tag{12}$$

Two other conditions that must be satisfied for the flow to represent reality. They are the no-penetration condition at the body surface

$$\vec{V} \cdot \vec{n} = 0 \quad \text{on } S \tag{13}$$

where $\bar{n}$ is the normal to the surface, and the far-field condition

$$\vec{V}_d(\infty) = 0 \tag{14}$$

When a thin lifting surface is immersed in the flow, two additional conditions must be met. They are the conservation of circulation, Γ, in the wake as prescribed by the Theorem of Kelvin and Helmholtz

$$\frac{D\Gamma}{Dt} = 0 \tag{15}$$

and the Kutta Condition

$$\Delta C_p = 0 \tag{16}$$

along the edge where the wake joins the wing.

The method developed by Mracek models the body as a collection of triangular vortex sheets, the corners of which lie on the actual surface of the body. The vorticity distribution across the triangle is a linear combination of the vorticities at each corner, or "node." The wake is modeled as a lattice of constant strength vortex cores.

Chapter 2 of Mracek gives the full development of the matrix equations which are solved to obtain the vorticity at each node in the global reference frame. The system of equations is overconstrained and is solved by satisfying the divergenceless condition (Equation 12) exactly and minimizing the error in the no-penetration condition (Equation 13) via a least square method.

<u>The Triangular Element of Vorticity</u> (1:14ff)

The present method's approach in modeling the wake as a continuous vortex sheet is to parallel the methods used by Mracek in approximating the wing. This not only prevents "reinventing the wheel," but also allows taking advantage of subroutines already in place and validated for calculating induced velocities. For this reason, a summary of Mracek's methods follows.

The disturbance velocity given by Equation 11 is approximated by the sum of the velocities induced by each of the triangular elements. Figure 1 depicts one such element, all of which have a local x-axis defined along its longest edge and a local y-axis defined such that the coordinates (b,c) are both positive.

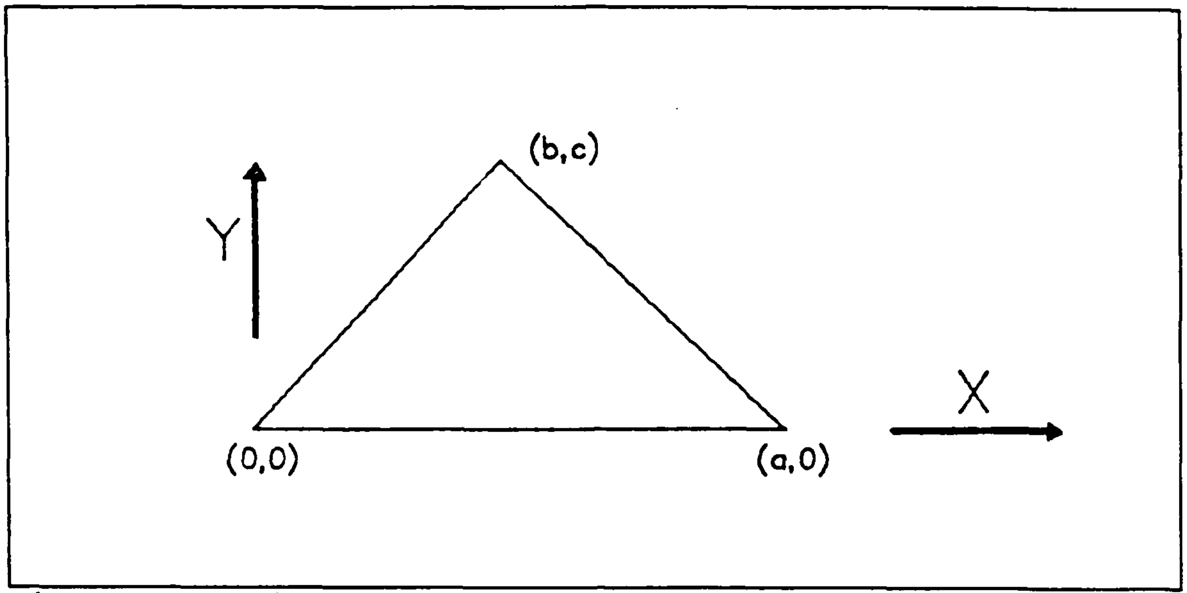

Figure 1 Triangular Element and Local Coordinate System (1:14)

The surface vorticity vector, $\vec{\gamma}$, of each element lies entirely within the plane of the element. That is, in the element's local frame,

$$\vec{\gamma} = \gamma_x \vec{i} + \gamma_y \vec{j} \qquad (17)$$

The vorticity across each element is approximated by a linearly varying combination of the vorticity vectors at each node.

$$\vec{\gamma} = (\gamma_{x1} f_1 + \gamma_{x2} f_2 + \gamma_{x3} f_3)\vec{i} + (\gamma_{y1} f_1 + \gamma_{y2} f_2 + \gamma_{y3} f_3)\vec{j} \qquad (18)$$

The γ_{xi} and γ_{yi} are the x- and y-components, respectively, of $\vec{\gamma}$ at corner "i" as shown in Figure 2. By convention, corner 1 is at the local frame's origin, corner 2 is at (a,0), and corner 3 is at (b,c). The f_i are basis functions which equal unity at their respective corners and zero at the other corners.

$$f_1 = a_1x + b_1y + 1$$
$$f_2 = a_2x + b_2y \qquad (19)$$
$$f_3 = a_3x + b_3y$$

where

$$a_1 = -1/a \qquad b_1 = (b-a)/ac$$
$$a_2 = 1/a \qquad b_2 = -b/ac \qquad (20)$$
$$a_3 = 0 \qquad b_3 = 1/c$$

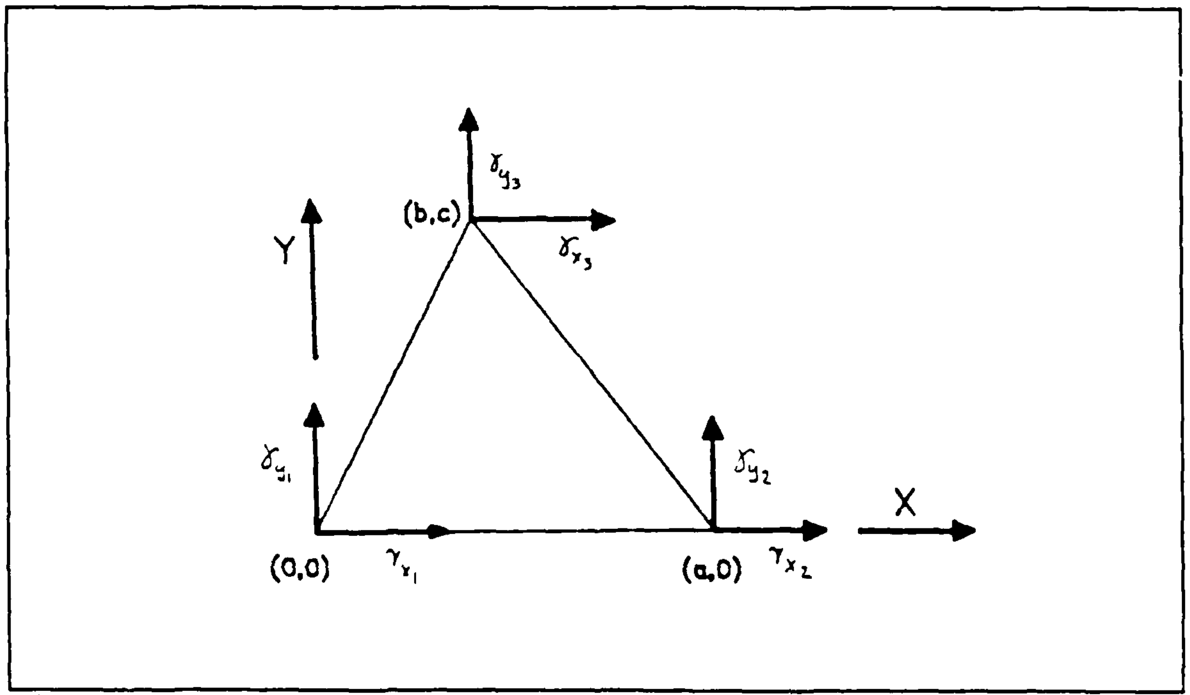

Figure 2 Triangular Element with Vorticity (1:56)

The divergence constraint (Equation 12) can now be written

$$\text{div}\ \vec{\gamma} = \frac{\partial \gamma_x}{\partial x} + \frac{\partial \gamma_y}{\partial y} = \sum_{i=1}^{3} a_i \gamma_{xi} + \sum_{i=1}^{3} b_i \gamma_{yi} = 0 \tag{21}$$

Edge Formulation (1:39)

For there to be non-zero vorticity perpendicular to the edge of a continuous vortex sheet, a variable strength vortex filament (core) lies along the edge to generate or capture the vorticity. The circulation (Γ) around this vortex core and the vorticity strength, γ, are related by (2:544)

$$\gamma(x) = - \frac{d\Gamma(x)}{dx} \tag{22}$$

where $\Gamma(x)$ is the circulation at position x along the length of the core and γ is the strength of the vortex sheet. The vorticity is perpendicular the edge core (i.e., $\gamma = \gamma_y$) and varies linearly on the triangular element as depicted in Figure 3. Along the edge,

$$\gamma_y = \frac{\gamma_{y2} - \gamma_{y1}}{d} x + \gamma_{y1} \tag{23}$$

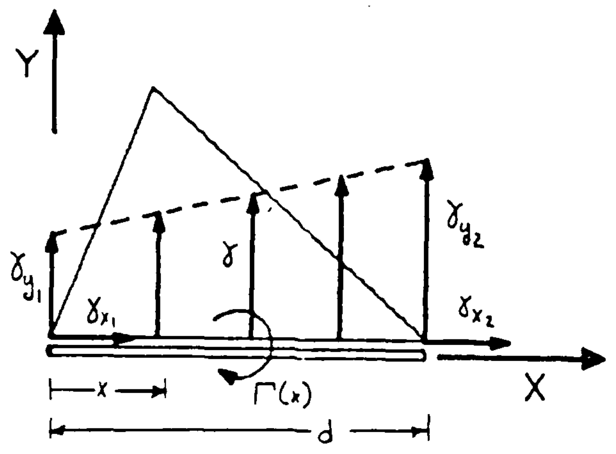

Figure 3 Edge Element with Associated Edge Core (1:40)

Integrating, the circulation of the core is

$$\Gamma(x) = \frac{\gamma_{y2} - \gamma_{y1}}{d} (x^2/2) + \gamma_{y1}\, x + G$$

$$= G_1\ (x^2/2) + G_2\, x + G_3 \qquad (24)$$

G_3 is an as yet unknown constant of integration. When modeling a body (or the wake) as a collection of triangular elements, the point where adjacent edge cores meet must exhibit continuous circulation strength. That is, for two adjacent elements, n and n+1, with asssociated edge cores meeting at point x_0,

$$\Gamma_n(x_0) = \Gamma_{n+1}(x_0) \tag{25}$$

Because of the relationship between circulation and vortex sheet strength, continuity through the first derivative is also enforced so that

$$\gamma_{y(n)}(x_0) = \gamma_{y(n+1)}(x_0) \tag{26}$$

These constraints actually add more equations than unknown constants of integration (G_3's), hence, the problem remains overconstrained.

Induced Disturbance Velocity

The model of a body immersed in the flow now has two components -- the triangular elements of linearly varying vorticity and the quadratically varying vortex filaments around the edges. The total disturbance velocity induced at any point in the flow is simply the sum of the velocities induced by each of these components.

$$\vec{V}_{d\ \text{body}} = \vec{V}_{d\ \text{elements}} + \vec{V}_{d\ \text{edge cores}} \tag{27}$$

The velocity induced by the triangular elements is given by Equation (11).

The velocity induced by the vortex filaments around the edge are calculated using the Biot-Savart law. For each edge filament, a local axis system is defined with the x-axis along the filament in the direction of the circulation. From Equation (24), the circulation along the filament is given by

$$\Gamma(x) = G_1 (x^2/2) + G_2 x + G_3 \tag{28}$$

The Biot-Savart law gives the velocity induced by the filament as

$$\vec{V}_c = \frac{1}{4\pi} \int \Gamma(x) \frac{d\vec{l} \times (\vec{r}-\vec{s})}{|\vec{r}-\vec{s}|^3} \tag{29}$$

where $d\vec{l}$ is a differential length along the filament

$$d\vec{l} = dx\, \vec{i} \tag{30}$$

when expressed in the local coordinate frame.

The Formation of the Wake

The wake is a vortex sheet emanating from the edge of the lifting surface. It must satisfy Equation (12) (the divergence condition), Equation (14) (far-field condition), and Equation (15) (conservation of circulation). The existing program generates the wake by impulsively starting the flow and taking snapshots of the developing wake at discrete time intervals. (1:65-69)

It must be understood at the outset that the intent of the present study is *not to change the method of generating and convecting the wake.* Rather, the intent here is to *change how the existing wake is modeled* once generated, for the purpose of calculating velocities induced by the vorticity in the wake. Any changes in induced velocities will, of course, alter the development of the wake, but the basic principles employed in developing the wake remain (1) the generation of edge cores on the wing, and (2) convecting them downstream at the local particle velocity.

To more clearly understand the focus of this study, consider a wing in an impulsively started flow. At the instant the flow is started, vorticity forms on the surface, and a vortex filament (core) with circulation develops around the edge of the wing. Not enough time has yet elapsed to move this circulation off the wing and into the flow. After an incremental time, the circulation is swept off the wing and downstream in the flow, and a vortex sheet stretches from the core to the edge from whence it originated. As this vorticity moves, its location relative to the wing, hence, the flow field on the surface of the wing is changing. At the end of the first time increment, there is a new distribution of vorticity on the surface, so the vortex sheet from the starting core to the surface of the wing is no longer compatible with the edge. To resolve the discontinuity, a new edge core is formed. The strength of this new core is exactly the negative of the strength of the starting core added to the strength of the core needed to generate the new surface vorticity in the absence of any wake.

At the start of the second time step, a new vortex distribution is formed on the surface because of the disturbance induced by the vorticity in the wake. Now edge cores are again formed to generate and capture the surface vorticity as in the first time step. Additionally, the negative of the starting vortex is collocated at the convecting edge to capture the surface vorticity of the wake sheet. Total point velocities are calculated for this configuration, and the new edge cores are convected into the wake along with the starting edge core and the vortex sheet which connects them. A new vortex sheet forms connecting the second edge core to the edge of the surface. This procedure is followed for all successive time steps.

In the existing program, the vorticity in the wake is not modeled as a continuous vortex sheet. Rather, the wake sheet is discretized into vortex cores of constant strength. The circulation strength chosen for this discrete core is the average circulation value of the convected variable strength vortex core. A four-sided ring of this circulation strength and composed of variable strength vortex cores connects the starting core to the convecting edge, satisfying the conservation of circulation -- the general unsteady vortex lattice method.

Figure 4 illustrates the formation of the wake. In Figure 4a, the flow is impulsively started, vorticity forms on the surface, and the associated edge cores have circulation. For clarity, only the trailing edge cores are shown to be convected, and they are shown slightly offset from the trailing edge. The average circulations of these edge cores are then calculated and convected off the trailing edge. A four-sided ring of circulation is formed connecting the starting vortex core to the trailing edge and a new trailing edge core is formed due to the change in

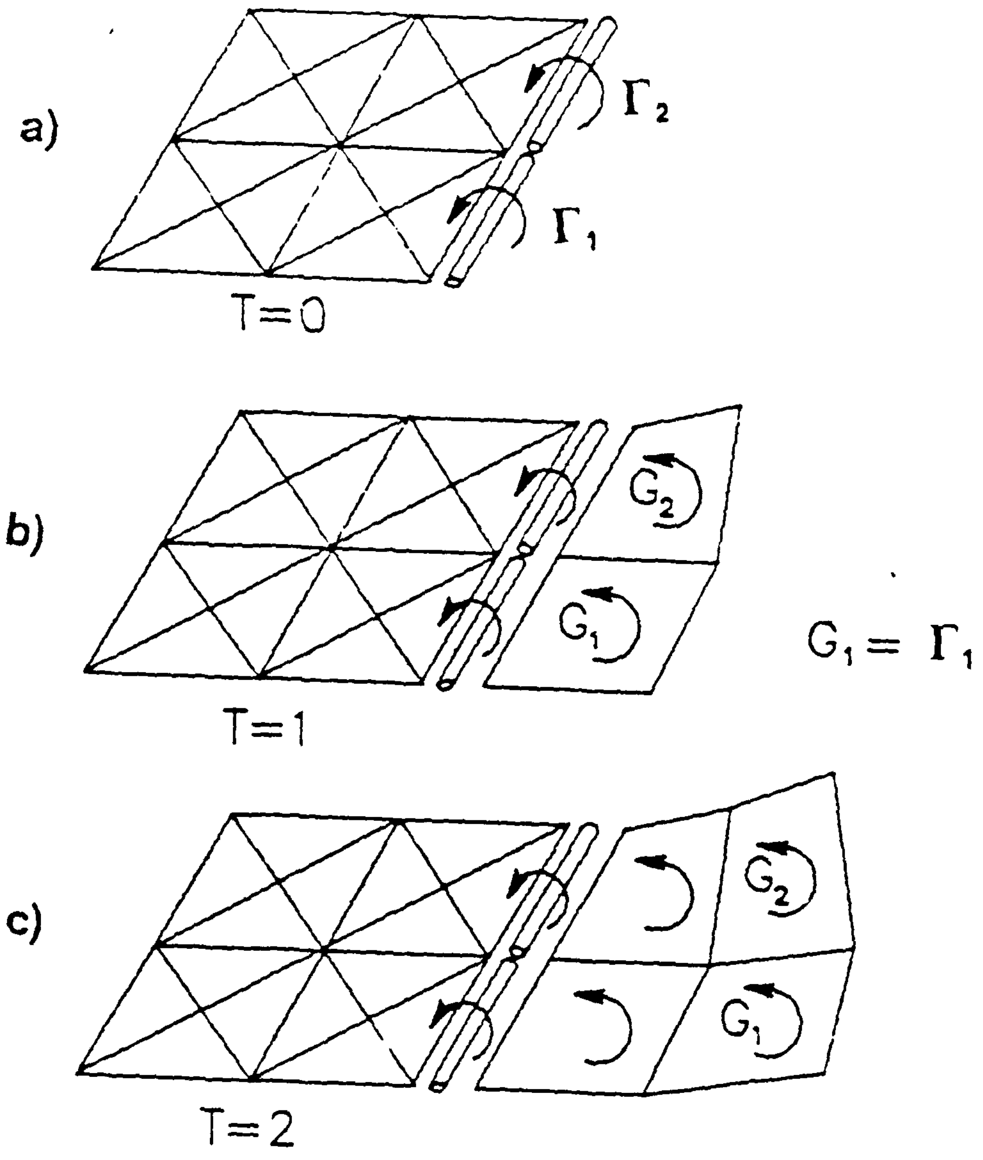

Figure 4 Development of Discrete Core Wake Lattice (1:68)

surface vorticity caused by the presence of the wake. This condition is shown in Figure 4b. The four corners of the ring are the "nodes" of the wake whose positions are updated each time step. Figure 4c shows the state after one more time step. In this method, only the average circulations and the updated positions of the nodes for the next time step are recorded to fully describe the wake.

Given the circulation of the variable strength core in Equation (28), the average circulation, Γ_{avg}, of a core of length d is found by

$$\Gamma_{avg} = \frac{\int_0^d G_1\ (x^2/2) + G_2\ x + G_3}{d}$$

$$= G_1\ (d^2/6) + G_2\ d + G_3 \qquad (31)$$

The requirement that the wake be single valued in its pressure distribution (Equation 16) is satisfied by convecting at the local particle velocity. For each of the nodes in the wake and along the trailing edge, the local velocity is calculated by

$$\vec{V}(t) = \vec{V}_{freestream} + \vec{V}_{d(body)} + \vec{V}_{d(wake)} \qquad (32)$$

and their new positions for the next time step are determined by the first-order difference formula

$$\vec{r}(t+\Delta t) = \vec{r}(t) + \vec{V}(t)\Delta t \qquad (33)$$

where Δt is the time increment between time steps.

The thrust of this work lies in the method of calculating $\vec{V}_{d(wake)}$ in Equation (32) above. The existing program, as stated earlier, uses the Biot-Savart law to determine the velocity induced by each of the constant strength cores of Γ_{avg} circulation and each of the variable strength filaments which form the four-sided circulation rings. $\vec{V}_{d(wake)}$, then, is simply the sum of these induced velocities. In the present method, with the wake modeled as a continuous vortex sheet, $\vec{V}_{d(wake)}$ is determined in the same manner as $\vec{V}_{d(body)}$, summing the velocities induced by triangular elements which approximate the vortex sheet, and the velocities induced by edge cores surrounding this new "body." The following example will not only illustrate this method, but will also reveal the motivation for the effort by demonstrating the advantages given in Chapter 1.

An Example of Equivalent Vortex Systems

The advantages given in Chapter 1 of using vortex panels over discrete cores in determining induced velocities can be demonstrated using a simple contrived example. This example was used by Mracek (1:72-74) to illustrate the validity of using discrete vortex cores in modeling the wake.

Consider a single row of the wake composed of four square elements of unit dimension as depicted in Figure 5. For simplicity, the entire row, modeled as a continuous vortex sheet, is planar with the vortex distribution as shown. For each segment, a local coordinate frame is defined with the positive x-direction in the direction of the circulation, and the positive y-direction toward the interior of the element. The basic equation relating circulation and vortex strength is

$$\frac{d}{dx} \Gamma(x) = - \gamma_y \tag{34}$$

For each of the ten discrete cores, the vorticity distribution is written in its local coordinate frame and integrated to obtain the equation for $\Gamma_i(x)$. For segment number 1,

$$\gamma_{y1} = -2x - 1 \tag{35a}$$

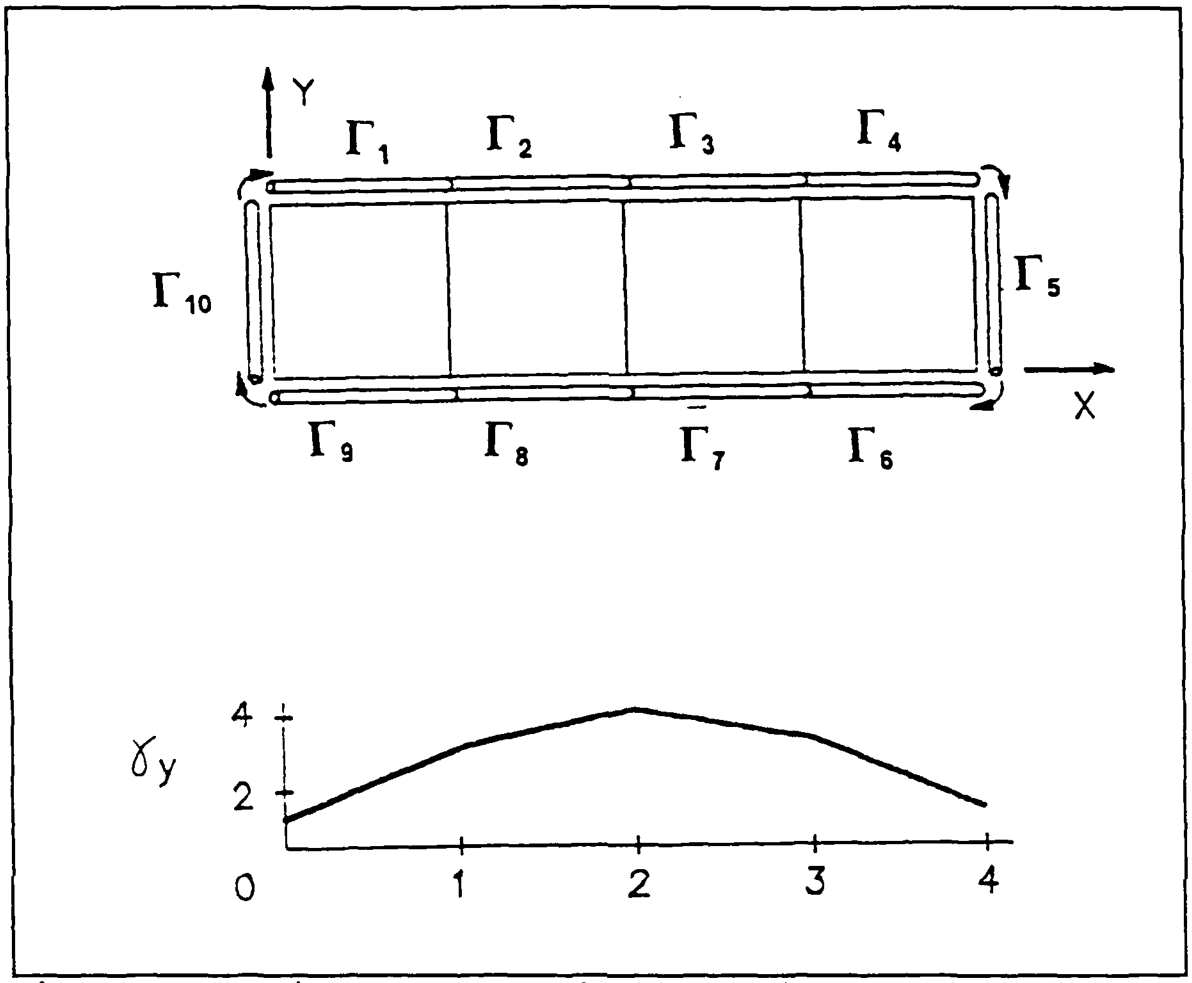

Figure 5 Continuous Vortex Sheet Example (1:72)

Integrating,

$$\Gamma_1(x) = 2(x^2/2) + x + K_1 \tag{35b}$$

where K_1 is an unknown constant of integration. Similarly,

$$\gamma_{y2} = -x - 3, \quad \Rightarrow \quad \Gamma_2(x) = 1(x^2/2) + 3x + K_2 \tag{36}$$

$$\gamma_{y3} = x - 4 \qquad \Gamma_3(x) = -1(x^2/2) + 4x + K_3 \tag{37}$$

$$\gamma_{y4} = 2x - 3 \qquad \Gamma_4(x) = -2(x^2/2) + 3x + K_4 \tag{38}$$

Note that for $\Gamma_5(x)$, since all vorticity is in the global y-direction, in the local coordinate frame,

$$\gamma_{y5} = 0 \qquad \Gamma_5(x) = K_5 \tag{39}$$

Continuing around,

$$\gamma_{y6} = 2x + 1 \qquad \Gamma_6(x) = -2(x^2/2) - x + K_6 \tag{40}$$

$$\gamma_{y7} = x + 3 \qquad \Gamma_7(x) = -1(x^2/2) - 3x + K_7 \tag{41}$$

$$\gamma_{y8} = -x + 4 \qquad \Gamma_8(x) = 1(x^2/2) - 4x + K_8 \tag{42}$$

$$\gamma_{y9} = -2x + 3 \qquad \Gamma_9(x) = 2(x^2/2) - 3x + K_9 \tag{43}$$

$$\gamma_{y10} = 0 \qquad \Gamma_{10}(x) = K_{10} \tag{44}$$

Now the circulation around the edge must be continuous so that

$$\Gamma_{10}(1) = \Gamma_1(0) \quad \Rightarrow \quad K_{10} = K_1 \tag{45}$$

$$\Gamma_1(1) = \Gamma_2(0) \quad \Rightarrow \quad 2 + K_1 = K_2 \tag{46}$$

$$\Gamma_2(1) = \Gamma_3(0) \quad \Rightarrow \quad 7/2 + K_2 = K_3 \tag{47}$$

$$\Gamma_3(1) = \Gamma_4(0) \quad \Rightarrow \quad 7/2 + K_3 = K_4 \tag{48}$$

$$\Gamma_4(1) = \Gamma_5(0) \quad \Rightarrow \quad 2 + K_4 = K_5 \tag{49}$$

$$\Gamma_5(1) = \Gamma_6(0) \quad \Rightarrow \quad K_5 = K_6 \tag{50}$$

$$\Gamma_6(1) = \Gamma_7(0) \quad \Rightarrow \quad -2 + K_6 = K_7 \tag{51}$$

$$\Gamma_7(1) = \Gamma_8(0) \quad \Rightarrow \quad -7/2 + K_7 = K_8 \tag{52}$$

$$\Gamma_8(1) = \Gamma_9(0) \quad \Rightarrow \quad -7/2 + K_8 = K_9 \tag{53}$$

$$\Gamma_9(1) = \Gamma_{10}(0) \quad \Rightarrow \quad -2 + K_9 = K_{10} \tag{54}$$

This yields nine independent equations in the ten K_i unknowns. To get the tenth equation, impose the additional constraint that

$$K_5 = -K_{10} \tag{55}$$

Solving this system of linear equations for K_i, the circulation distributions are then

$$\Gamma_1(x) = x^2 + x - 11/2 \tag{56}$$

$$\Gamma_2(x) = x^2/2 + 3x - 7/2 \tag{57}$$

$$\Gamma_3(x) = -x^2/2 + 4x \tag{58}$$

$$\Gamma_4(x) = -x^2 + 3x + 7/2 \tag{59}$$

$$\Gamma_5(x) = 11/2 \tag{60}$$

$$\Gamma_6(x) = -x^2 - x + 11/2 \tag{61}$$

$$\Gamma_7(x) = -x^2/2 - 3x + 7/2 \tag{63}$$

$$\Gamma_8(x) = x^2/2 - 4x \tag{64}$$

$$\Gamma_9(x) = x^2 - 3x - 7/2 \tag{65}$$

$$\Gamma_{10}(x) = -11/2 \tag{66}$$

The equivalent discrete vortex core model can now be calculated as shown in Figure 6 where

$$g_1 = \hat{\Gamma}_1 = \hat{\Gamma}_9 \tag{67}$$

$$g_2 = \hat{\Gamma}_2 = \hat{\Gamma}_8 \tag{68}$$

$$g_3 = \hat{\Gamma}_3 = \hat{\Gamma}_7 \tag{69}$$

$$g_4 = \hat{\Gamma}_4 = \hat{\Gamma}_6 \tag{70}$$

with

$$\hat{\Gamma}_i = \int \Gamma_i(x)\ dx \tag{71}$$

For this example, the length of each segment is unity, so

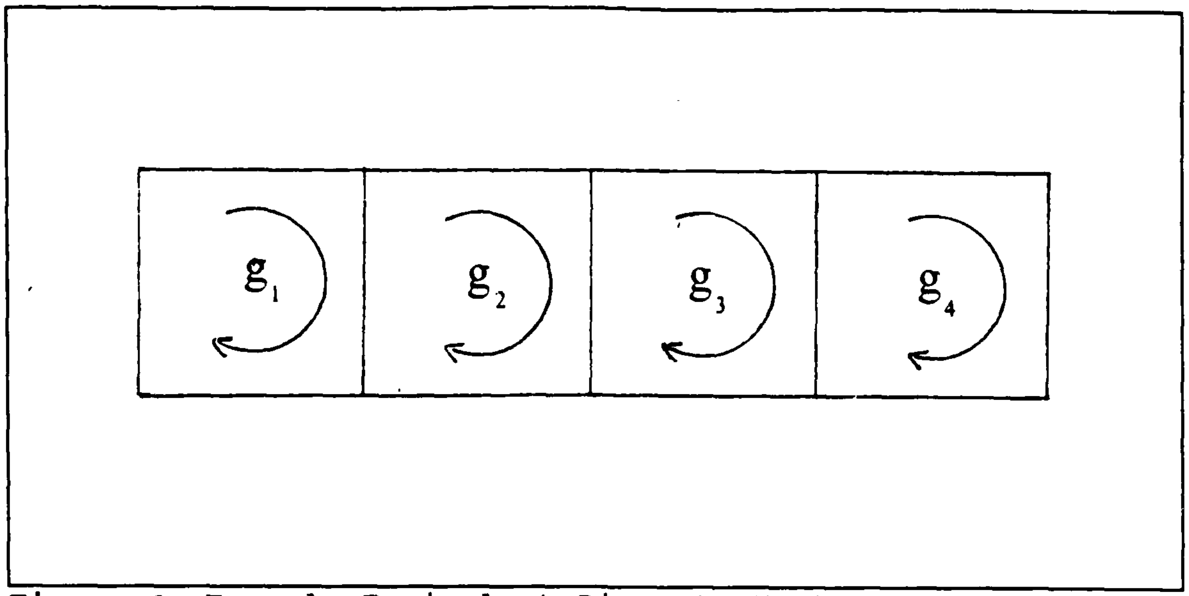

Figure 6 Example Equivalent Discrete Vortex Core System

the limits of integration are zero to one. Solving for each $\hat{\Gamma}_i$ (hence, the g_i's)

$$g_1 = -14/3, \quad g_2 = -11/6, \quad g_3 = 11/6, \quad g_4 = 14/3 \qquad (72)$$

Figure 7 illustrates the equivalent discrete vortex core system.

The discrete core and continuous sheet representations are compared by analysis of the velocity induced in a planar region normal to the plane of the vortex sheet. Figure 8 shows the velocity field around the continuous vortex sheet as calculated by the existing program as well as the velocity field for the equivalent discrete core arrangement. Note that the induced velocity fields are nearly identical except for very near the sheet where the continuous sheet

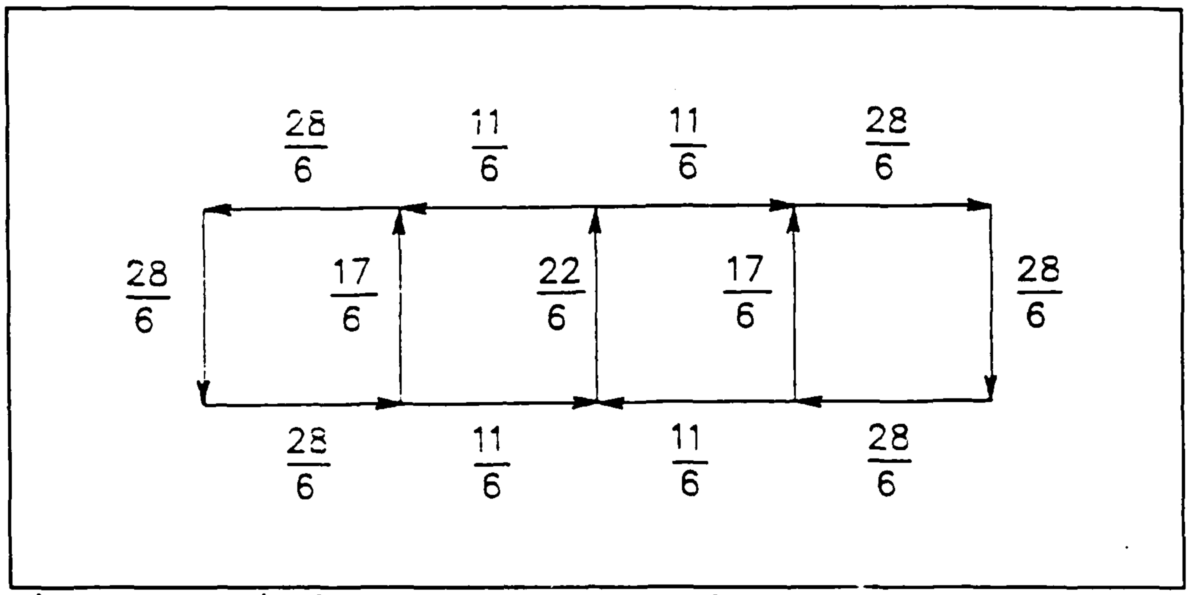

Figure 7 Equivalent Constant Strength Vortex Core Arrangement (1:73)

representation gives a much more uniform induced velocity distribution. This demonstrates two of the statements made in Chapter 1 -- (1) the continuous sheet yields induced velocities more closely approximating reality, and (2) this advantage is only realized in close proximity to the vortex sheet.

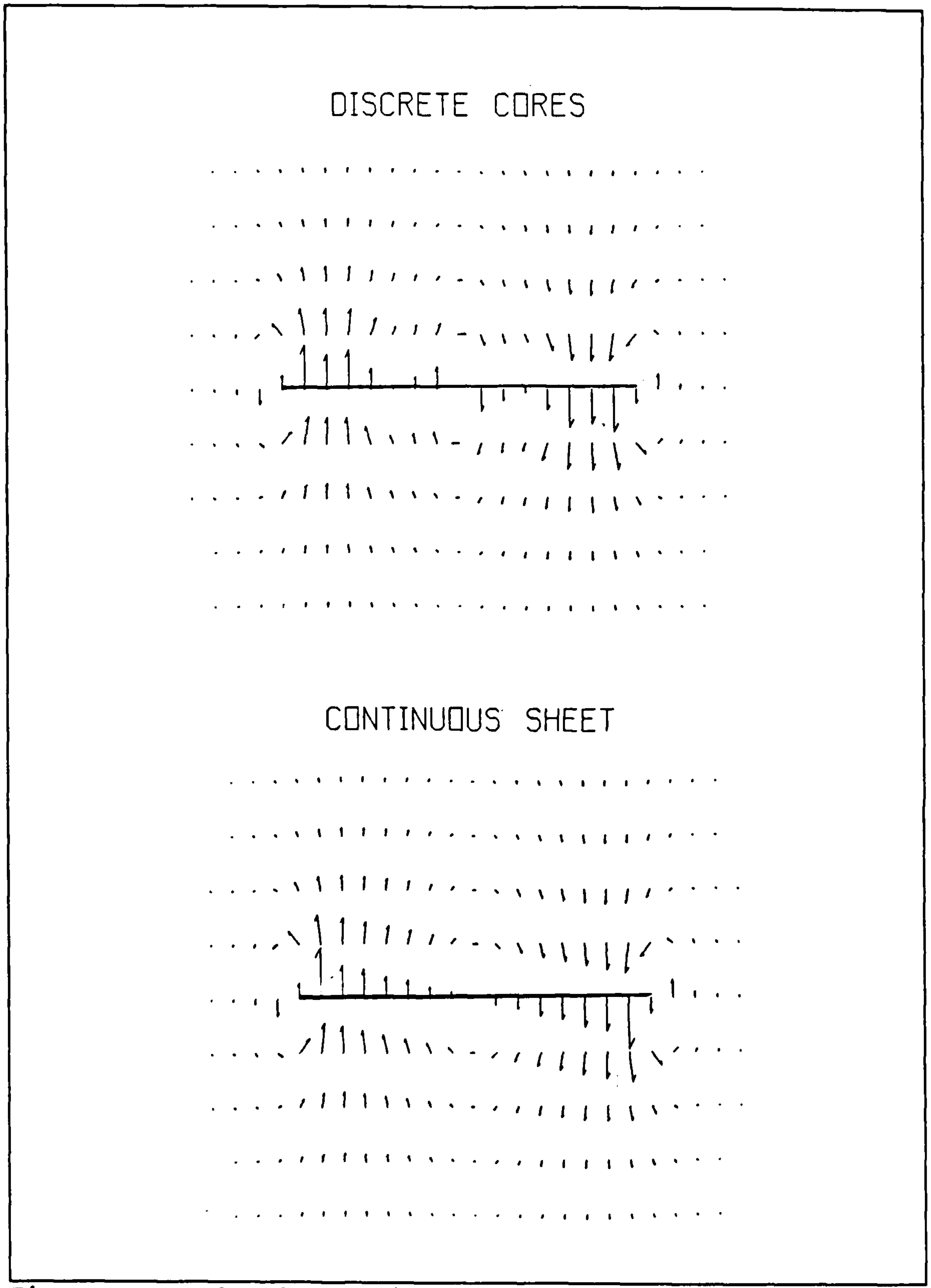

Figure 8 Induced Velocity Fields of Comparative Example (1:74)

III. Modeling the Continuous Vortex Sheet

In addition to illustrating the motivation for this study, the previous example also revealed that the existing program *already has* much of the capability required to handle a wake modeled as a continuous vortex sheet. Subroutines are already in place which can calculate the disturbance velocities induced by a continuous sheet, given that the information required to describe that continuous sheet is available. After implementation, the primary difference between the program of this effort and the old version is in the method of determining the disturbance velocities induced by the wake. The new version sums the influence of triangular elements which approximate the wake and the edge cores with circulation that surround them, just as the existing program does for the wing.

The objective of this effort, then, is to devise a scheme for discretizing the wake into triangular elements of vorticity that approximate the wake sheet and account for the variable strength cores which surround it; calculating and recording all the characteristics required by the velocity subroutines used for the wing. The algorithm which models the wing serves as a pattern for this effort, but modifications must be made since the wake is continually growing and distorting. In particular, the conservation of circulation is maintained in the existing program by way of the constant strength Γ-average values which convect with the four sided rings of circulation. The vortex strength of the sheet inside that ring is directly related to that constant strength circulation. The first obstacle, then, is to somehow distribute the vorticity strength over an increasing area as the four sided ring grows with time.

Redistributing the Vorticity Over an Increasing Area

Figure 9 shows the convection of a typical edge core of length d off the trailing edge of the wing. As the endpoints of the filament are convected, their updated positions for the next time step are now separated by a distance of $d + \Delta d$. Associated with this filament are the G_i coefficients of Equations 28 and 31 used for calculating Γ_{avg}. An algorithm must be devised to correct these coefficients for the change in filament length while maintaining the constant Γ_{avg}.

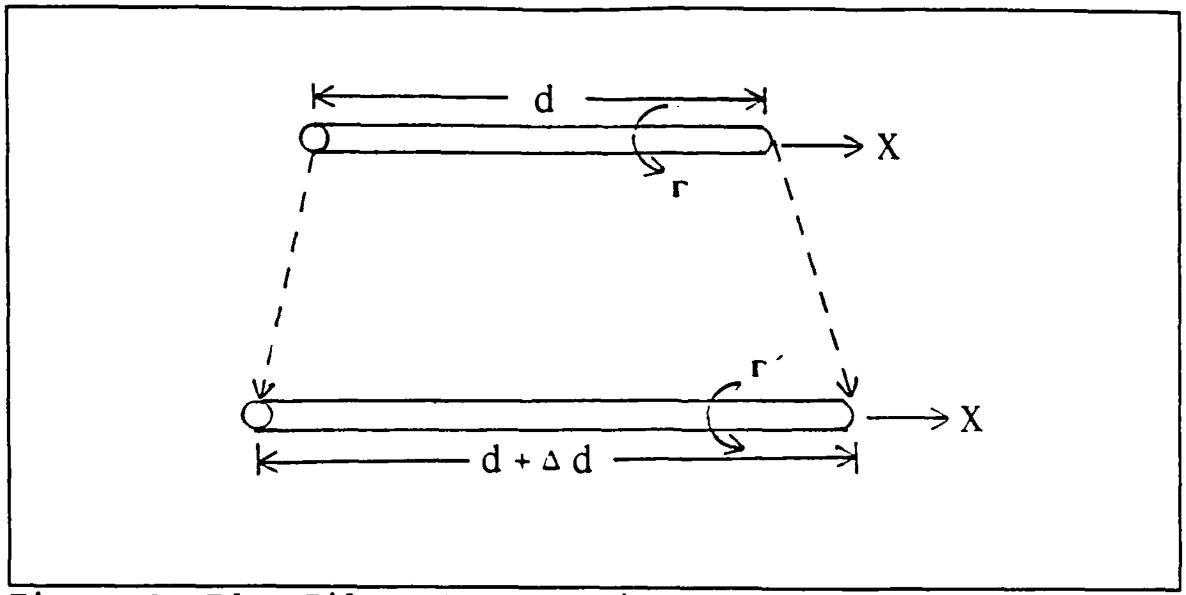

Figure 9 Edge Filament Convection

That is, if denoting the coefficients corrected for growth as primed,

$$\begin{aligned}\Gamma_{avg} &= G_1(d^2/6) + G_2(d/2) + G_3 \\ &= G_1'[(d+\Delta d)^2/6] + G_2'[(d+\Delta d)/2] + G_3' \qquad (74)\end{aligned}$$

The first attempt was to simply equate the terms of like powers in the Γ_{avg} equation. This yielded the relationships

$$G_1' = G_1 [d/(d+\Delta d)]^2 \qquad (75)$$

$$G_2' = G_2 [d/(d+\Delta d)] \qquad (76)$$

$$G_3' = G_3 \qquad (77)$$

Upon running this relationship, it was found that there were marked discontinuities at the nodes where adjacent filaments meet which violated Equation (25) (circulation strength node compatibility) and Equation (26) (vorticity strength node compatibilty). The method made no consideration for node compatibility, solving for each filament independently, and, in retrospect, actually forced the discontinuity at the nodes. A new approach was taken to consider the entire row of the wake generated in a time step, maintaining the average strength and node compatibility while correcting for growth and distortion.

System of Linear Equations in G-Primes. To solve for the coefficients (G-primes) corrected for wake growth, a system of linear equations in G-primes was constructed for a row of the wake n vortex filaments wide. Figure 10 shows a simple row of only n=2 elements to demonstrate the conditions enforced in this system of equations. The first condition is that Γ_{avg} remains constant.

$$\Gamma'_{i\ avg} = \Gamma_{i\ avg} \tag{78}$$

The second and third conditions enforce node compatibility in both $\Gamma(x)$ and $\gamma(x)$.

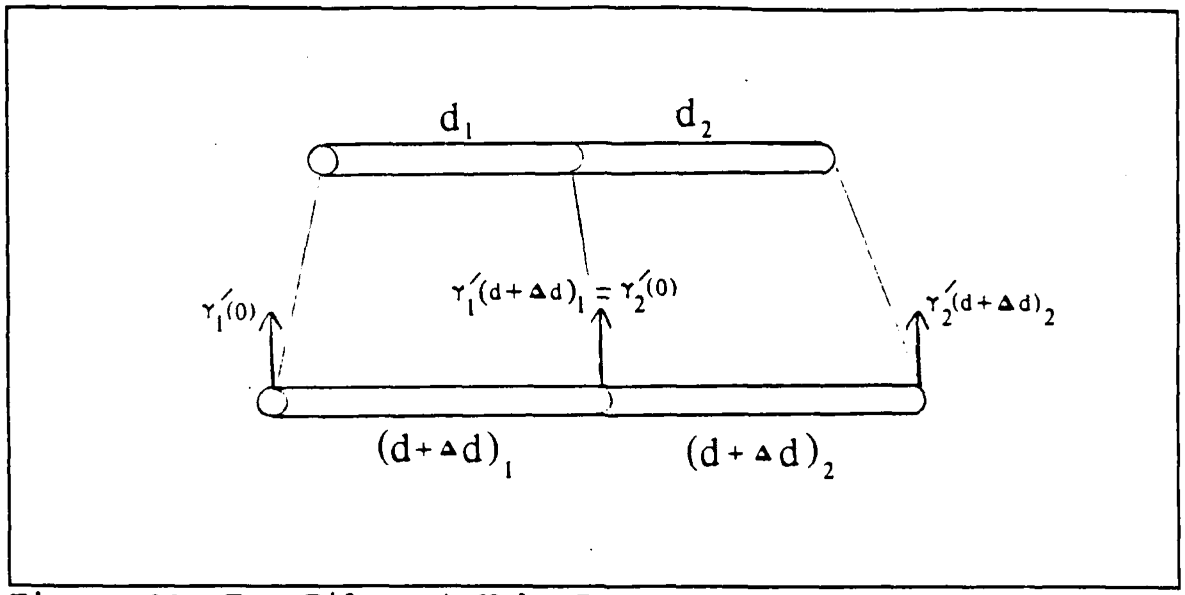
d_1
d_2
$\gamma_1'(0)$
$\gamma_1'(d+\Delta d)_1 = \gamma_2'(0)$
$\gamma_2'(d+\Delta d)_2$
$(d+\Delta d)_1$
$(d+\Delta d)_2$

Figure 10 Two Filament Wake Row

$$\Gamma'_i(d+\Delta d) = \Gamma'_{i+1}(0) \tag{79}$$

$$\gamma'_i(d+\Delta d) = \gamma'_{i+1}(0) \tag{80}$$

The final conditions are referred to as "end conditions." The two end cores are constant strength (not variable in their local x) such that for the left end,

$$\Gamma'_1(0) = \Gamma_1(0) \quad \Rightarrow \quad G'_{31} = G_{31} \tag{81}$$

and for the right end (the n-th filament),

$$\Gamma'_n(d+\Delta d) = \Gamma_n(d) \tag{82}$$

For each convected edge core, then, there are three G'_i

unknowns. For each, the available equations include the Γ_{avg} equation and, except for the last core, the two node compatibility equations. Together with the two end conditions, a system of 3n equations has been constructed in 3n unknowns.

This system of equations is represented in matrix form in the program so that a standard simultaneous equation solver could be employed. By loading the equations into the matrix in a specific order, a banded matrix results with bandwidth seven; a characteristic which can be exploited by equation solvers to decrease computational time. The first equation loaded into the matrix is the left end condition (Equation 81). Then stepping through each filament from left to right, the Γ_{avg} equation (Equation 78) and the two node compatibility equations (Equations 79 and 80) are loaded. For the last (n-th) filament, only the Γ_{avg} equation and the right end condition (Equation 82) are loaded. The resulting coefficient matrix is banded to ± three elements of the main diagonal, and is solved by a subroutine MSOLV (3) which takes advantage of the banded characteristic. For a wake row n-filaments wide, the matrix equation is constructed as

$$\begin{bmatrix}
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \cdots & 0 \\
\frac{(d+\Delta d)_1^2}{6} & \frac{(d+\Delta d)_1}{2} & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \cdots & 0 \\
\frac{(d+\Delta d)_1^2}{2} & (d+\Delta d)_1 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \cdots & 0 \\
(d+\Delta d)_1 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \cdots & 0 \\
0 & 0 & 0 & \frac{(d+\Delta d)_2^2}{6} & \frac{(d+\Delta d)_2}{2} & 1 & 0 & 0 & 0 & 0 \cdots & 0 \\
0 & 0 & 0 & \frac{(d+\Delta d)_2^2}{2} & (d+\Delta d)_2 & 1 & 0 & 0 & 0 & 0 \cdots & 0 \\
0 & 0 & 0 & (d+\Delta d)_2 & 1 & 0 & 0 & 0 & -1 & 0 \cdots & 0 \\
\vdots & & & & & & & & & & \\
0 & 0 & 0 & 0 & 0 \cdots & & 0 & 0 & \frac{(d+\Delta d)_n^2}{6} & \frac{(d+\Delta d)_n}{2} & 1 \\
0 & 0 & 0 & 0 & 0 \cdots & & 0 & 0 & \frac{(d+\Delta d)_n^2}{2} & (d+\Delta d)_n & 1
\end{bmatrix}
\begin{bmatrix}
G'_{11} \\ G'_{21} \\ G'_{31} \\ G'_{12} \\ G'_{22} \\ G'_{32} \\ G'_{33} \\ \vdots \\ G'_{2n} \\ G'_{3n}
\end{bmatrix}
=
\begin{bmatrix}
G_{31} \\ \Gamma_{avg1} \\ 0 \\ 0 \\ \Gamma_{avg2} \\ 0 \\ 0 \\ \vdots \\ \Gamma_{avgn} \\ \Gamma_n(d_n)
\end{bmatrix} \quad (83)$$

The Fortran algorithm which builds and solves this matrix equation is in subroutine CONVE (Appendix A). A test case was run with a rectangular wing with unity aspect ratio at 5 degrees angle of attack. This case convects twelve edge cores each time step, four from each side and four from the trailing edge. The run was taken to 60 time steps which ensured a steady state condition in the wake. The G-prime coefficients were calculated and stored using Equation (83) in the old version of the program, but they were not used in any calculations affecting wake convection . The resulting coefficients were analyzed by plotting the core strength (Equation 28) and the vortex sheet strength (Equation 34) for the trailing edge and for wake rows 5, 10, 15, and 20 of the wake. Figure 11 demonstrates that the core strength remains smooth and continuous as the wake spreads. Figure 12, the vortex sheet strengths, is even more revealing. As the wake spreads, not only is node compatibility maintained, but the strength of the sheet is being redistributed over the increased area as reflected in the shallowing of the plot as the wake span increases. That is, the area under the curve is remaining constant. The odd function type symmetry about the midspan indicates that spanwise and directional symmetry is also maintained. It is concluded that the method to redistribute vortex sheet strength as the wake expands is successful. The corrected G_i coefficients are required later by the subroutines which calculate disturbance velocities induced by the wake.

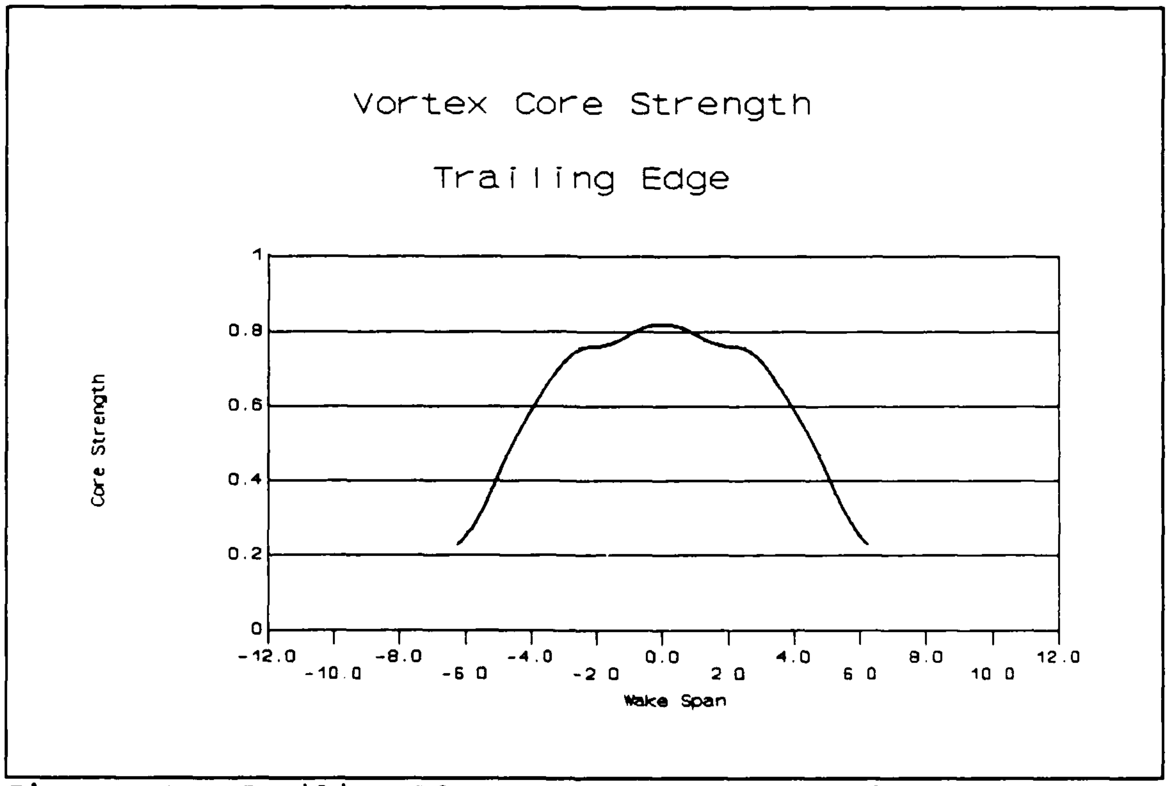

Figure 11a Trailing Edge Vortex Core Strength

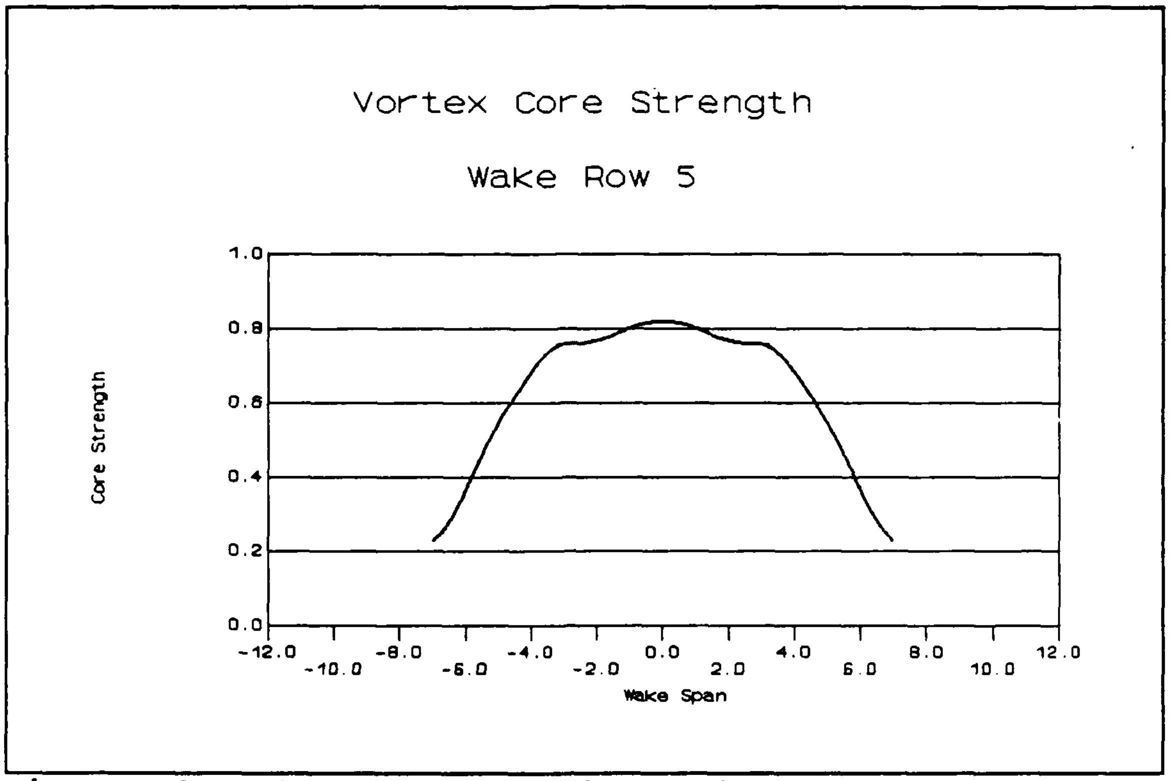

Figure 11b Vortex Core Strength -- Wake Row 5

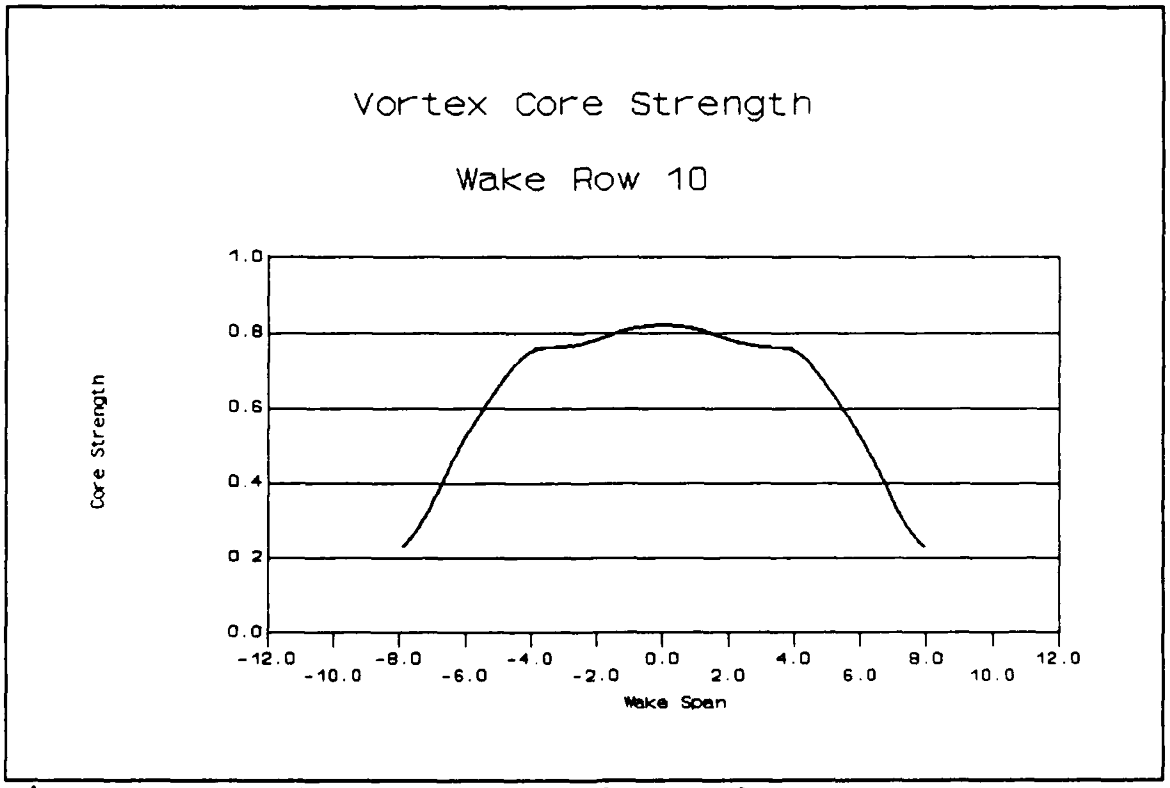

Figure 11c Vortex Core Strength -- Wake Row 10

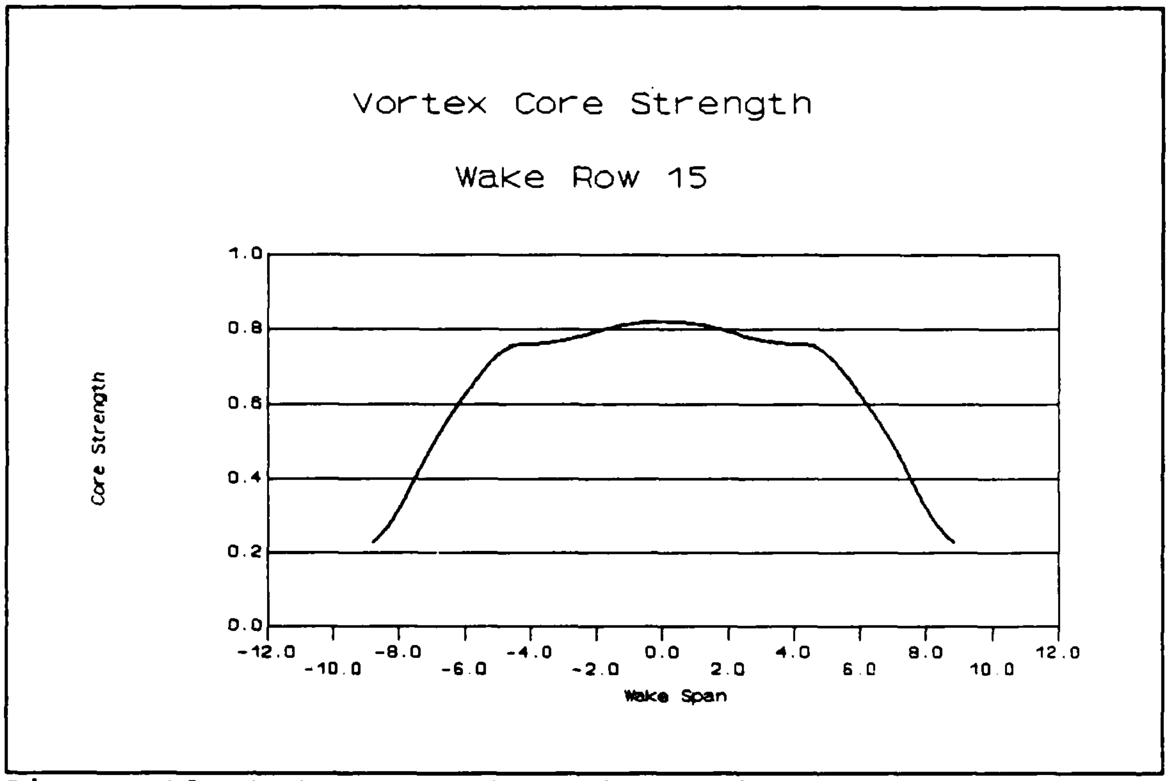

Figure 11d Vortex Core Strength -- Wake Row 15

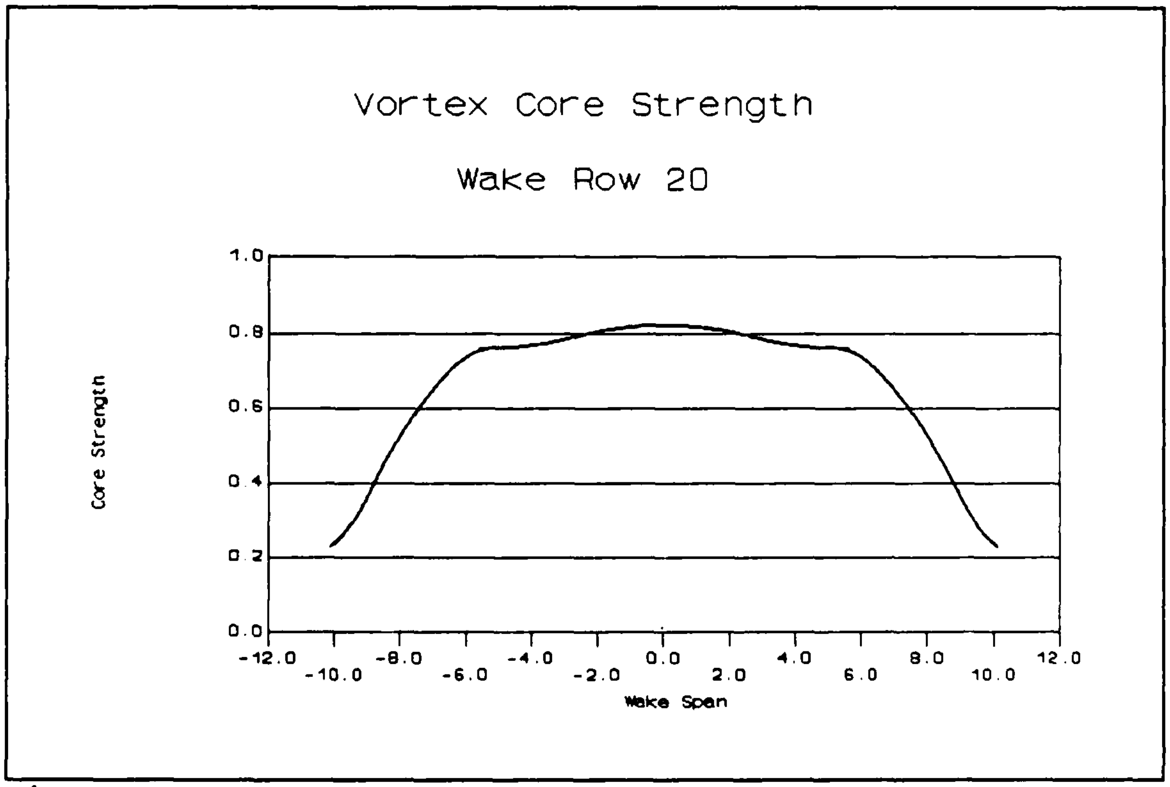

Figure 11e Vortex Core Strength -- Wake Row 20

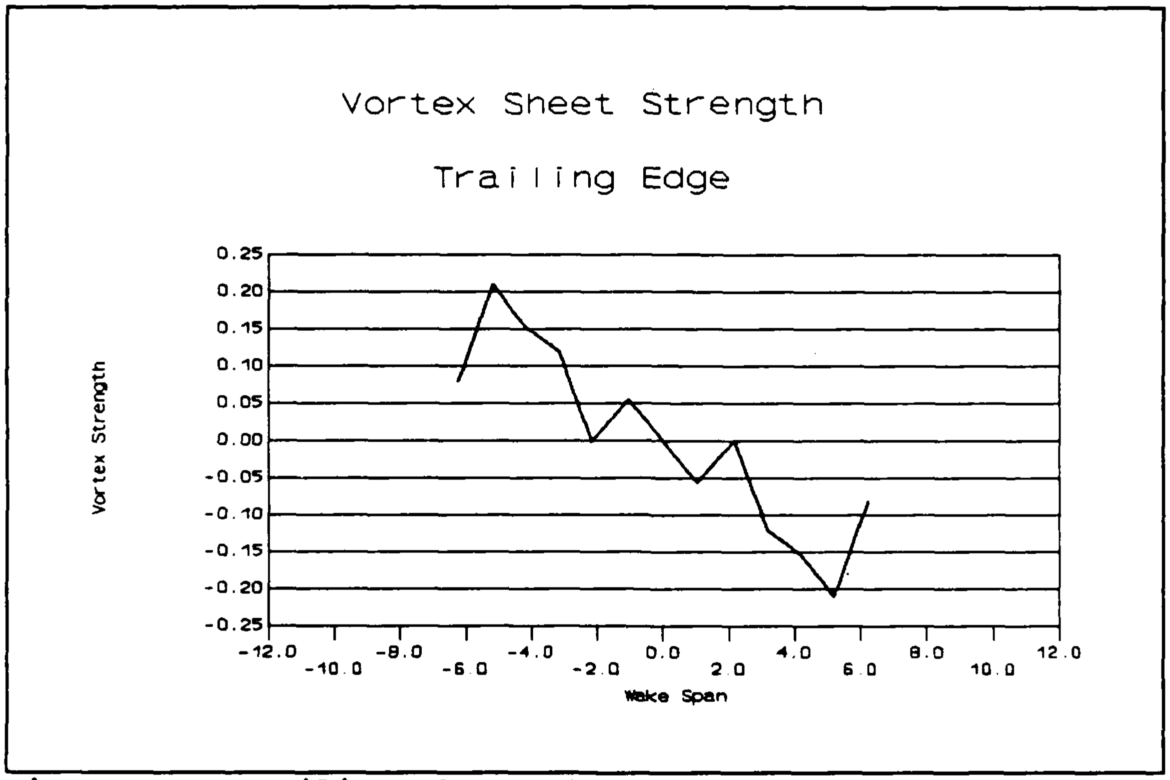

Figure 12a Trailing Edge Vortex Sheet Strength

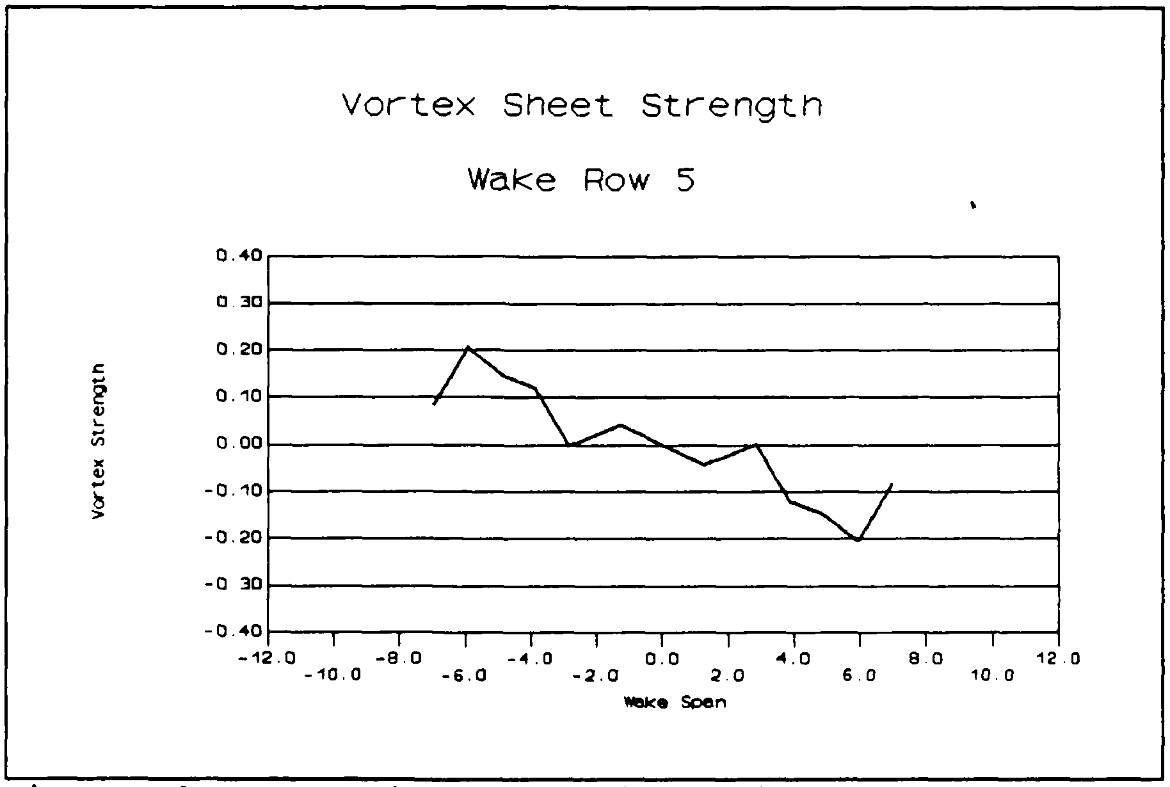

Figure 12b Vortex Sheet Strength -- Wake Row 5

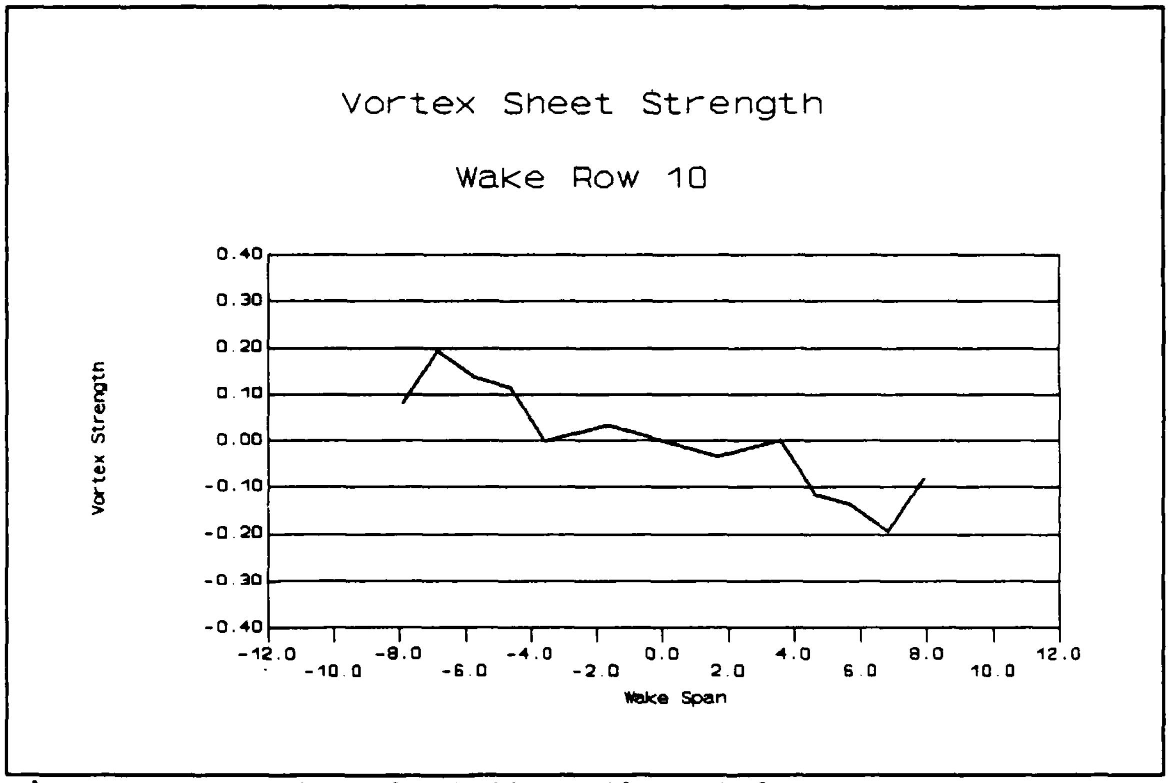

Figure 12c Vortex Sheet Strength -- Wake Row 10

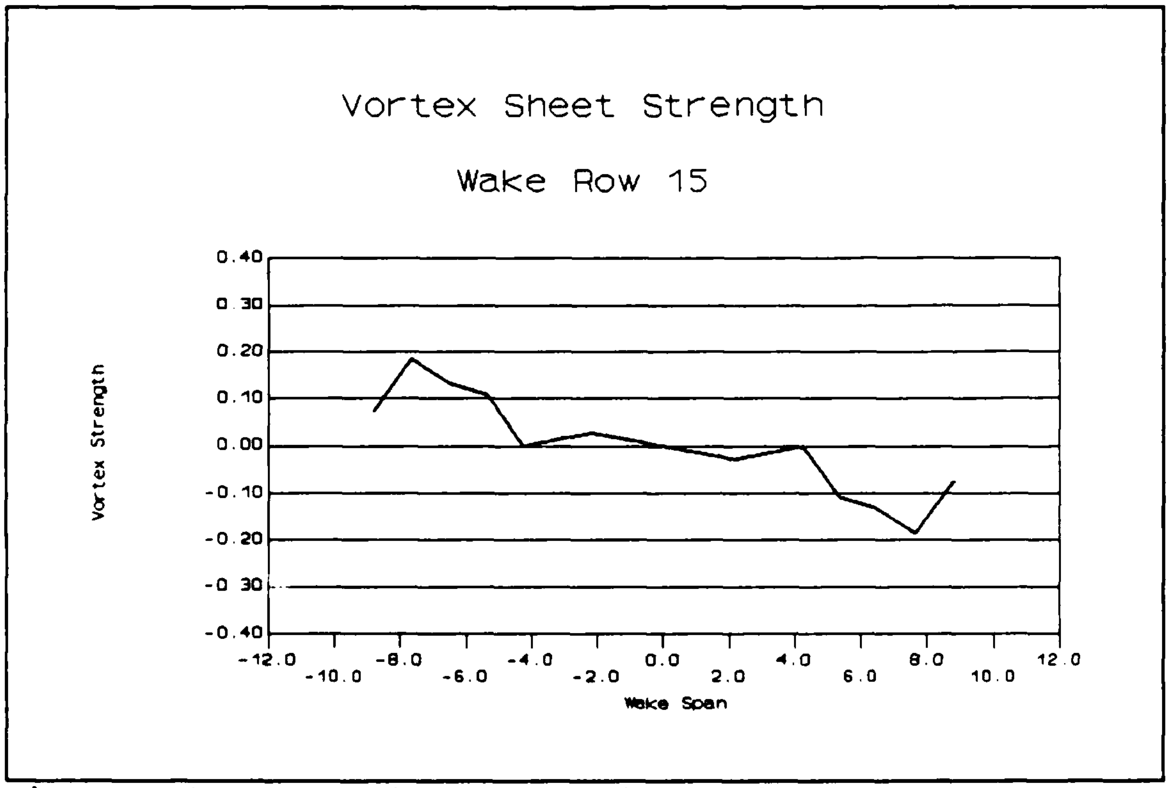

Figure 12d Vortex Sheet Strength -- Wake Row 15

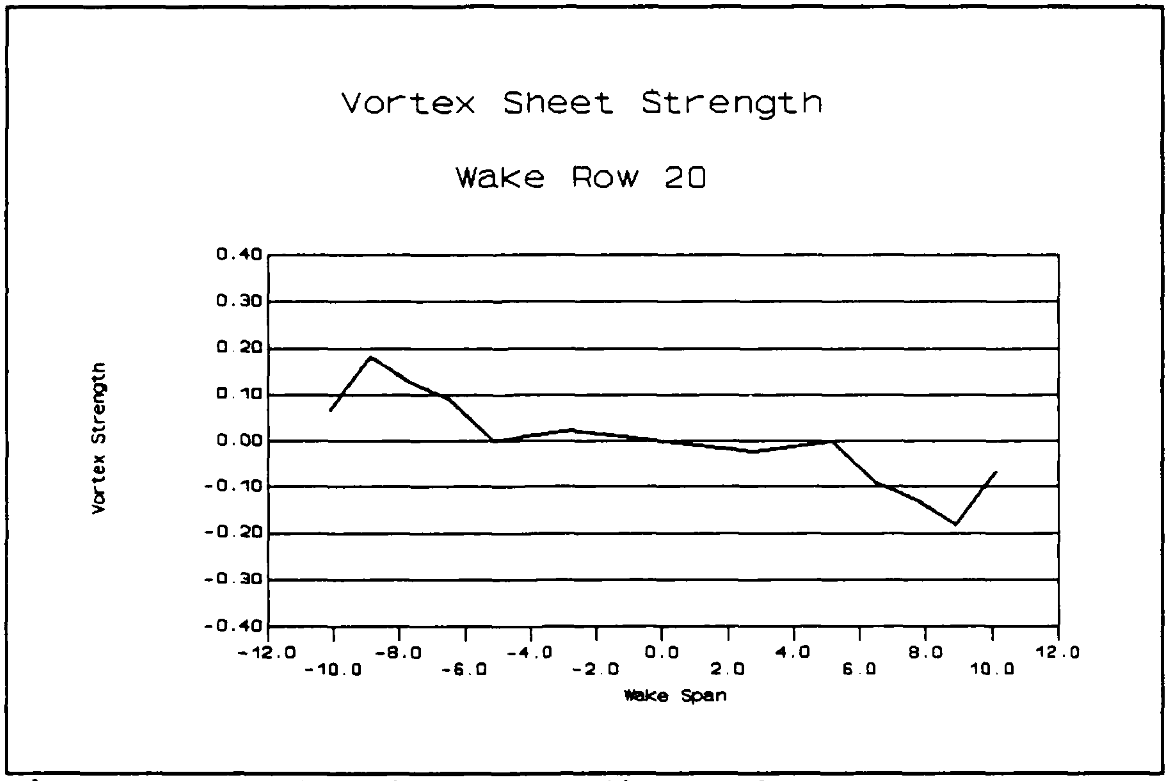

Figure 12e Vortex Sheet Strength -- Wake Row 20

<u>Discretizing the Wake</u>

Having successfully accounted for wake growth in redistributing the vortex sheet strength, a design is developed for discretizing the wake into the triangular elements of vorticity. In addition to expanding, the four sided rings of circulation are not planar as they convect. In fact, they may become quite distorted, especially in the outer regions of the span where the wake sheet is rolling up. The scheme that was chosen approximates the four sided element with four triangles that share a common vertex at the "midpoint" of the quadrilateral. This "midpoint" is simply the average of the x-, y-, and z-coordinates of the four corner nodes. It is envisioned that the four triangles will provide the flexibility required to adequately approximate the quadrilateral even in cases of extreme twisting as may be encountered in the rolling up of the wake. An example of the discretization scheme is shown in Figure 13, depicting two wake rows four edge cores wide, each surrounded by its own edge cores. For clarity, this figure shows a nonexistent gap between the two rows. Hereafter, the upstream side of the row is referred to as the "1-position" which correlates with subscripting methods in the Fortran. Similarly, the downstream side is referred to as the "2-position."

In order to properly account for the influence of the wake when calculating induced velocities, each row of the wake must be treated as a new "body." This is due to the fact that until steady state is reached, the strength of the surrounding edge cores on the upstream (1-position) side of a given row is not the same as the strength of the cores on the downstream (2-position) side of the previous wake row (although they *are* collocated). Therefore, the G-prime coefficient matrix equation must be constructed and solved

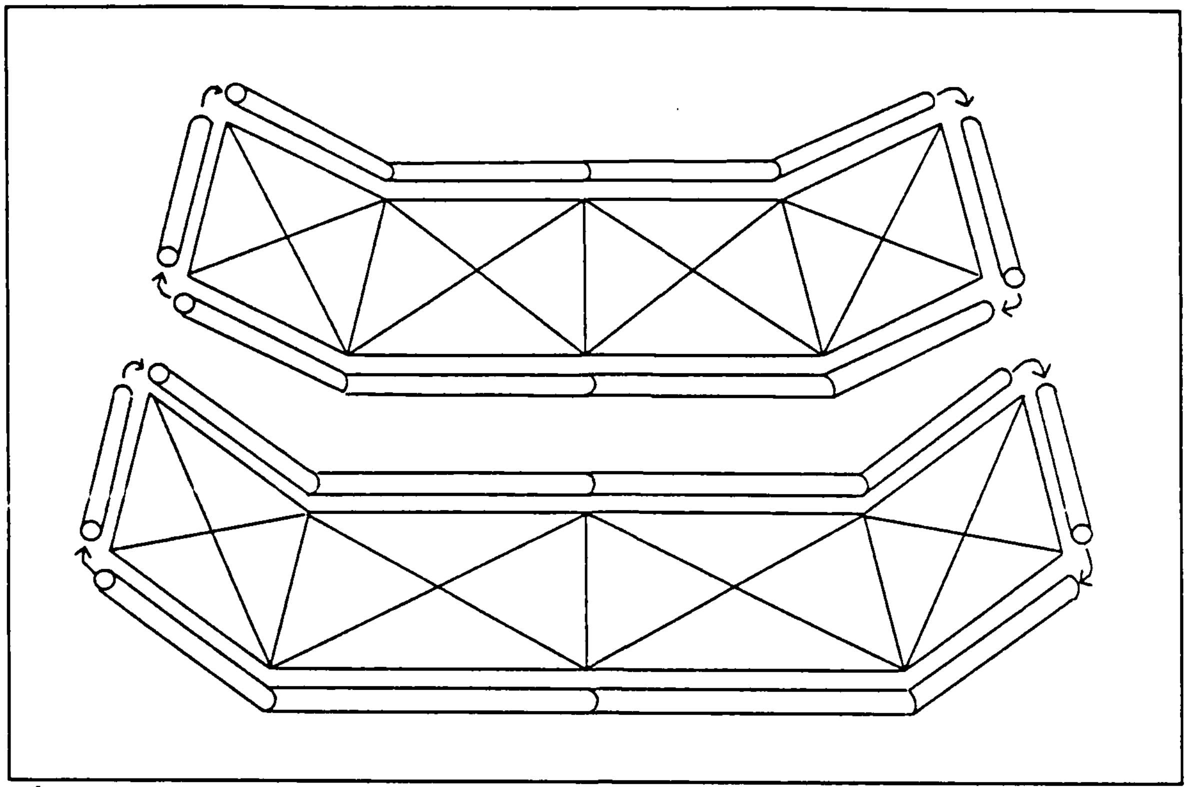
Figure 13 Example Wake Row Discretization

separately for both the 1- and 2-position filaments for each wake row. With this design established, all that remains is to provide the induced velocity calculating subroutines with the characteristics they require to define these "additional bodies" in the flow.

Characteristics of the Wake Rows

The existing subroutines that calculate induced disturbance velocities for a continuous vortex sheet require several characteristic values which define its geometry and vorticity. Once again, each row of the wake is treated as an individual body or continuous sheet. Since the program already modeled the wake as discrete cores, much of the required information was already available or could be readily calculated from existing values to make the equivalent continuous vortex system. The modeling of the continuous system looks much like the inverse of the process in the example at the end of Chapter 1, which transformed the wake into the equivalent discrete system. Because of the nature of the discretization scheme, some assumptions and simplifications are made.

An important feature of this work is that the characteristics of the wake rows are appended onto the characteristic arrays defining the wing. This allows using the same variable names, hence, existing subroutines to calculate induced velocities. Minimal bookkeeping effort (by node or element number subscript) is required to associate characteristic values with their respective continuous sheet. The wake characteristics and any assumptions or simplifications made in defining them follow. Except where otherwise noted, the characteristic values are calculated and stored in subroutine CONSHT (Appendix B).

Position Coordinates. The updated positions of each wake node were already calculated in subroutine CONVE in the existing program. These are determined using the first-order finite difference formula given in Equation (33), and are appended each time step to the arrays that define the geometry of the wing.

Element Orientation. Several arrays define the orientation of each element. A direction cosine matrix relating each element's local frame to the global frame is computed and stored using an algorithm identical to that used for the elements that approximate the wing. For elements which have one side on an outside edge, the direction of circulation is specified for the edge core along that side. A vector normal to each element is calculated by cross product of the local x-axis with the side from (a,0) to (b,c), and an unit normal is calculated for each node by averaging the normals of every triangular element that shares that node. Finally, Mracek's velocity subroutines require a "c-factor" (1:22-26), calculated using Equation 2.4-20 of his dissertation (Ref.1). The c-factor is a measure of co-planarity of elements sharing a given node, and is calculated for each vertex of each triangle as

$$c = \frac{[\omega_x^2 + \omega_y^2 + \omega_z^2]^{1/2}}{[\omega_x^2 + \omega_y^2 + \omega_z^2 + (\omega_z/n_z)^2]^{1/2}} \qquad (84)$$

where the ω's are the x-, y-, and z-components of the vorticity vector at the node expressed in the element's local frame, and n_z is the z-component of the average normal at the node expressed in the element's local frame.

Vorticity. Once the orientation of the elements is specified, the basis functions (Equations 19) for that element are computed. The vorticities at each node along the 1- and 2-positions are calculated using Equation (22) and the G_i coefficients which have been corrected for growth. The vorticities are calculated in the edge cores' local frames and transformed to global coordinates, but with one modification. Figure 14 shows a single quadrilateral ring in the wake. The vorticity at each of the four corners is calculated using Equation (22), but the direction is not defined perpendicular to the edge core as discussed in Chapter 2. Rather, to yield more accurate induced velocities directionally, the vorticity is rotated into the x-direction of the left and right triangular elements of the quadrilateral, more closely approximating the equivalent discrete core system. Finally, the vorticity vector of the newly defined "midpoint" node is calculated by averaging the global components of the four corner vorticities.

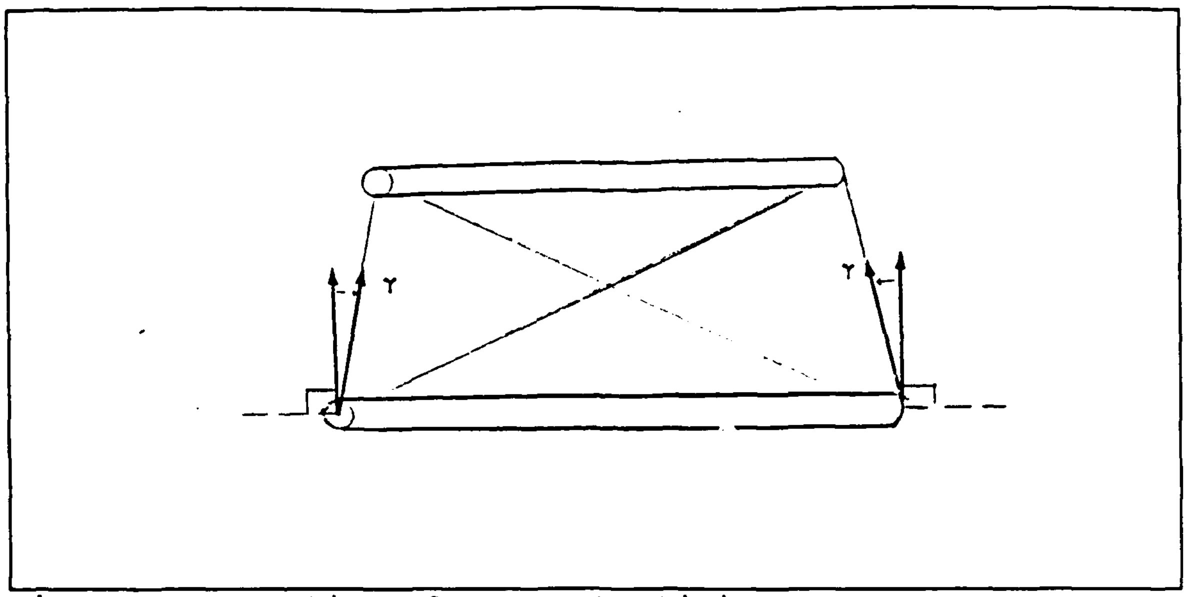

Figure 14 Rotation of Corner Vorticity Vectors

<u>Velocity Induced by the Wake</u>

The entire wake has now been modeled as a continuous vortex sheet, or more accurately, as several rows of continuous vortex sheets (each wake row is a continuous sheet). To determine the particle velocity at any point in the flow, the contribution of the wake is calculated for inclusion in Equation 32. The existing program called a subroutine which calculated wake induced velocities using the discrete core vortex system. The present effort replaced this subroutine with one nearly identical to the routine called for velocities induced by the body (wing). Subroutine VELWK is included in Appendix C.

IV. Results

Modeling the wake as a continuous vortex sheet yielded mixed results. Several computer simulations were attempted for unit aspect ratio rectangular and delta wings at 5, 10, 15, and 20 degrees angle of attack for comparison with results of Mracek's program. The simulations encountered numerical difficulties for the rectangular wing at 20 degrees angle of attack and for all angles of attack with the delta wing. Successful runs were made with the rectangular wing, however, at 5, 10, and 15 degrees. These results may be enough for analysis of the theory in general by comparing with results of the program that models the wake as discrete cores. It appears as though the theory may in fact be valid and feasible based on the comparisons as shown in Figures 15, 16, and 17. These figures show cross sections of the wake at several constant x-values for 5, 10, and 15 degrees angle of attack, respectively. Good correlation is demonstrated for these limited conditions, the most obvious difference being the amount of rolling up in the wake. The suspected causes of the numerical difficulties are discussed in the next chapter.

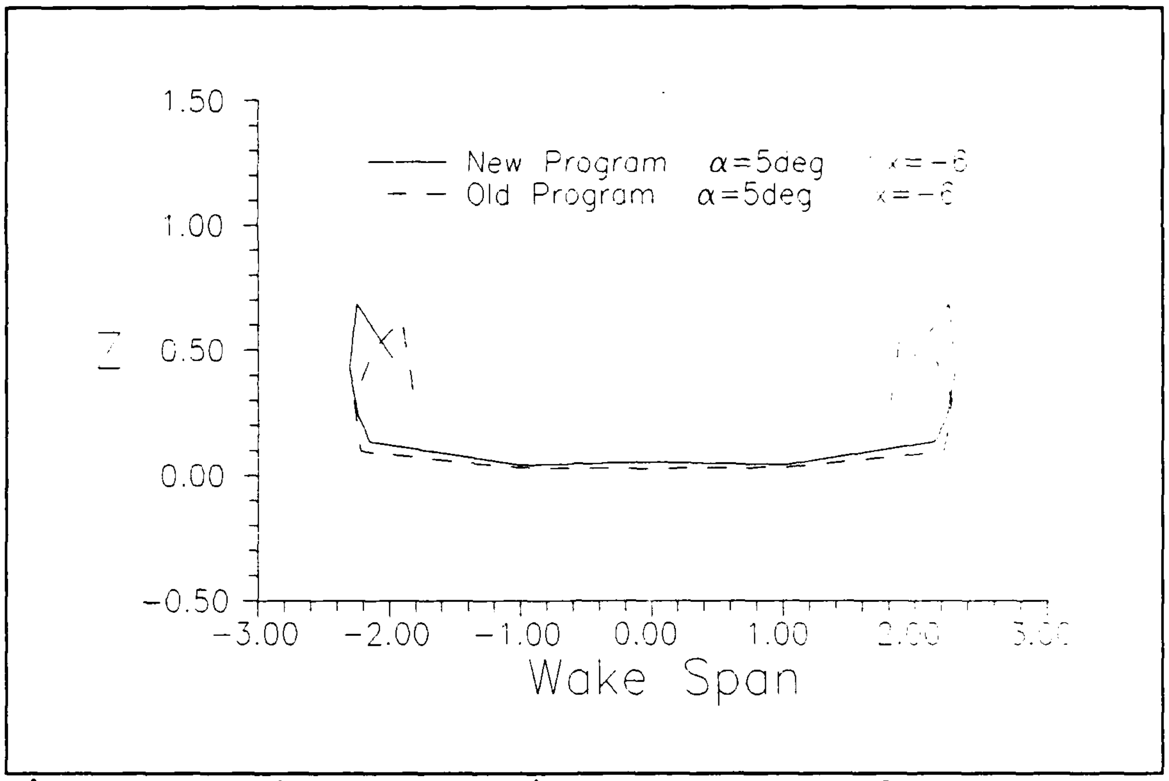

Figure 15a Wake Cross Sections (A.O.A. = 5 deg., x = -6)

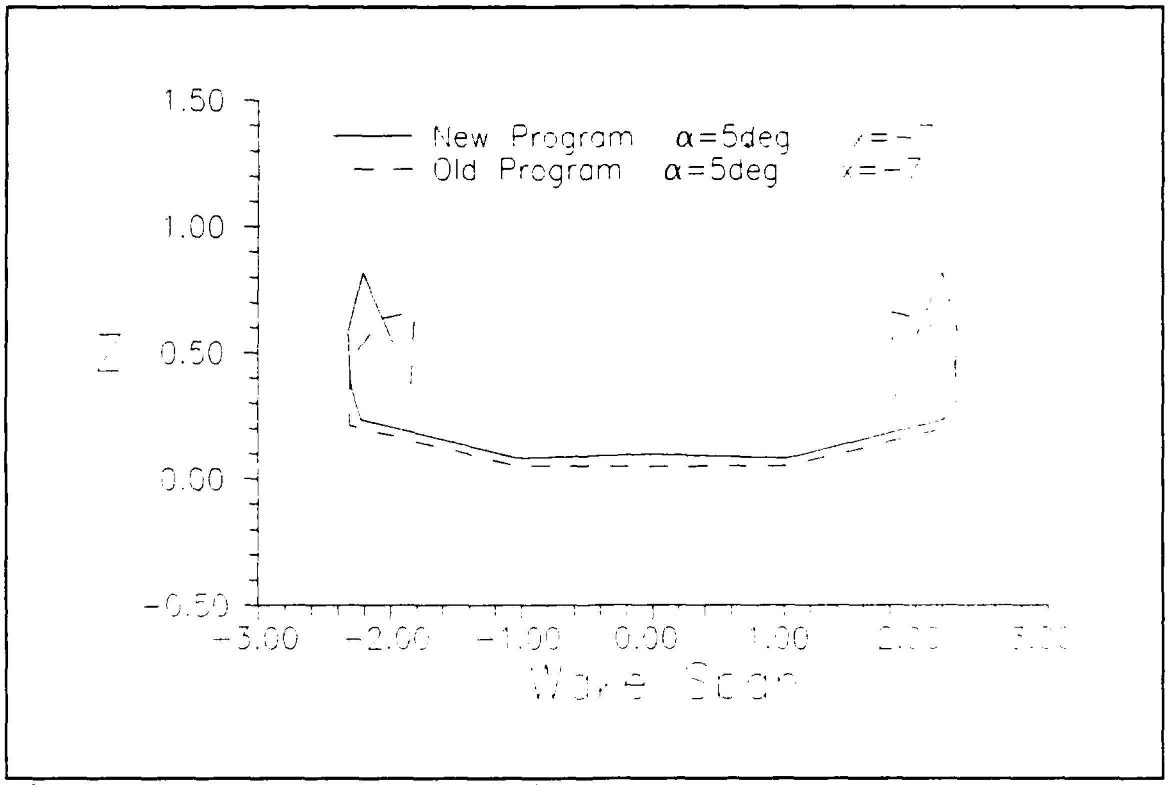

Figure 15b Wake Cross Sections (A.O.A = 5 deg., x = -7)

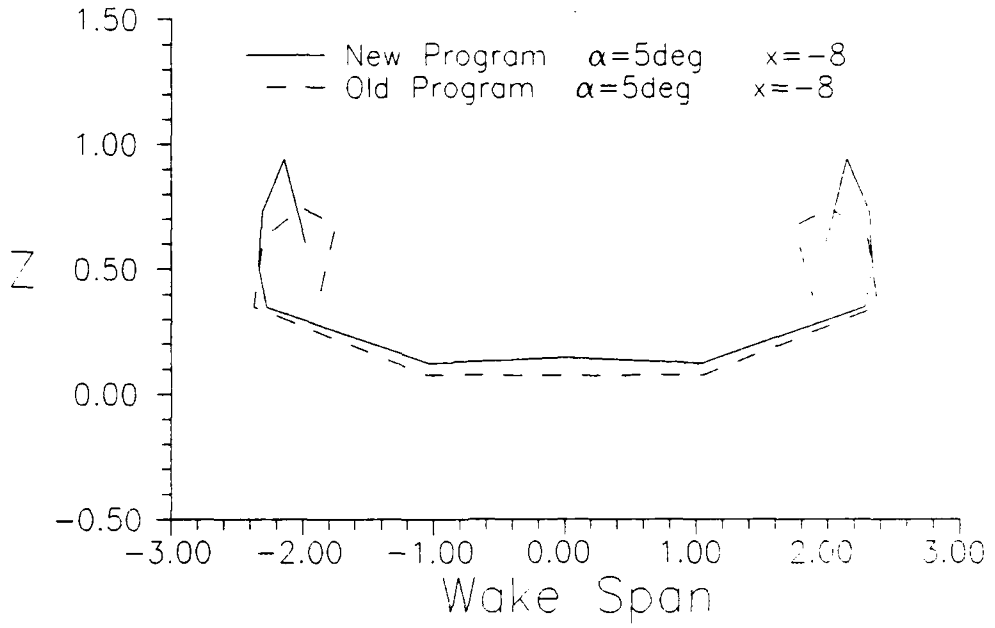

Figure 15c Wake Cross Sections (A.O.A. = 5 deg., x = -8)

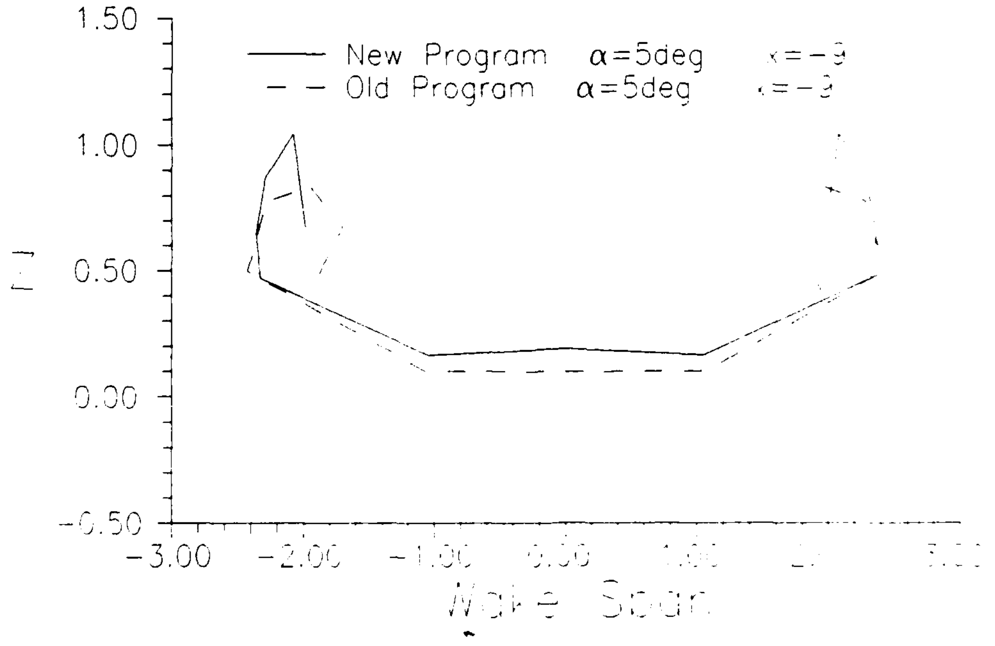

Figure 15d Wake Cross Sections (A.O.A. = 5 deg., x = -9)

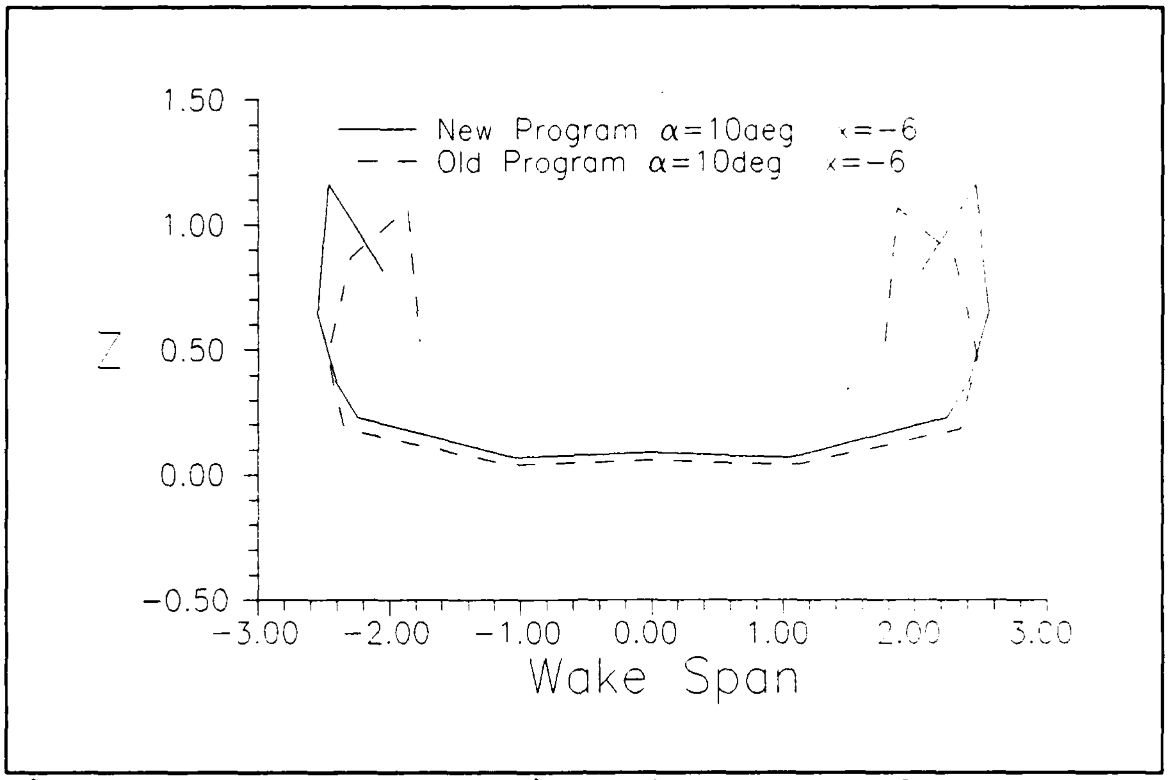

Figure 16a Wake Cross Sections (A.O.A. = 10 deg., x = -6)

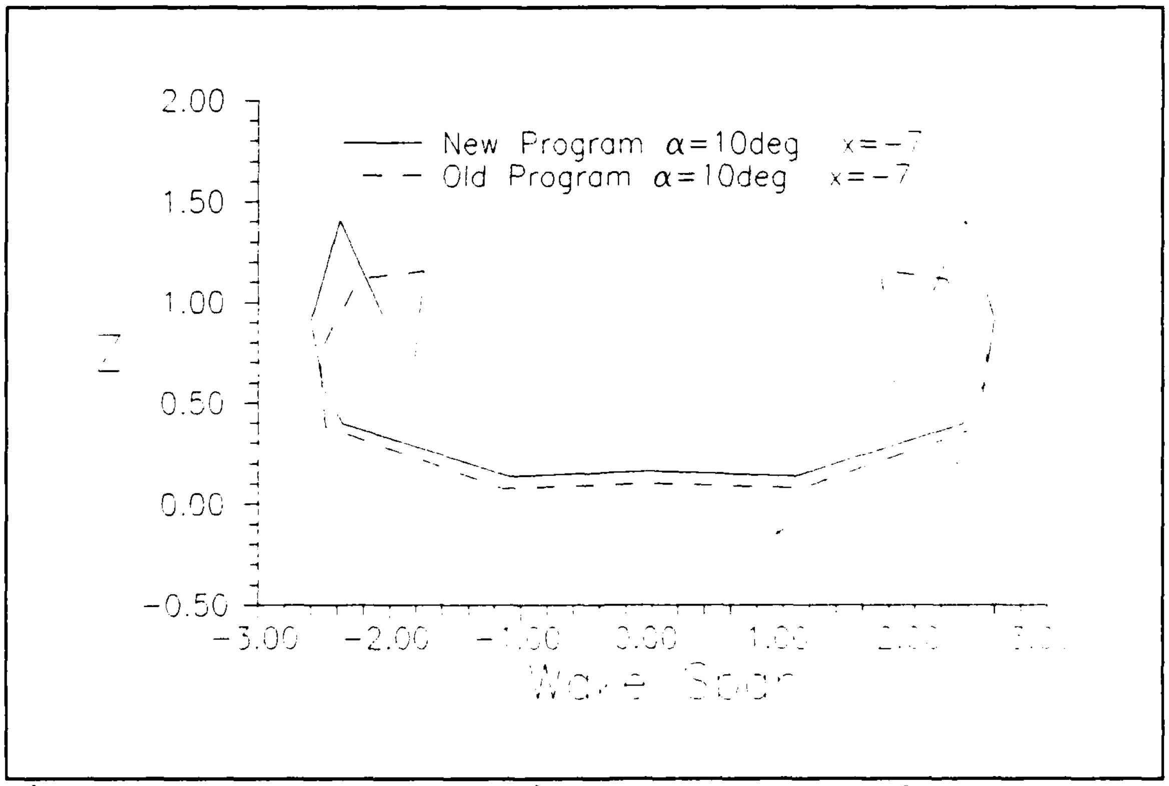

Figure 16b Wake Cross Sections (A.O.A. = 10 deg., x = -7)

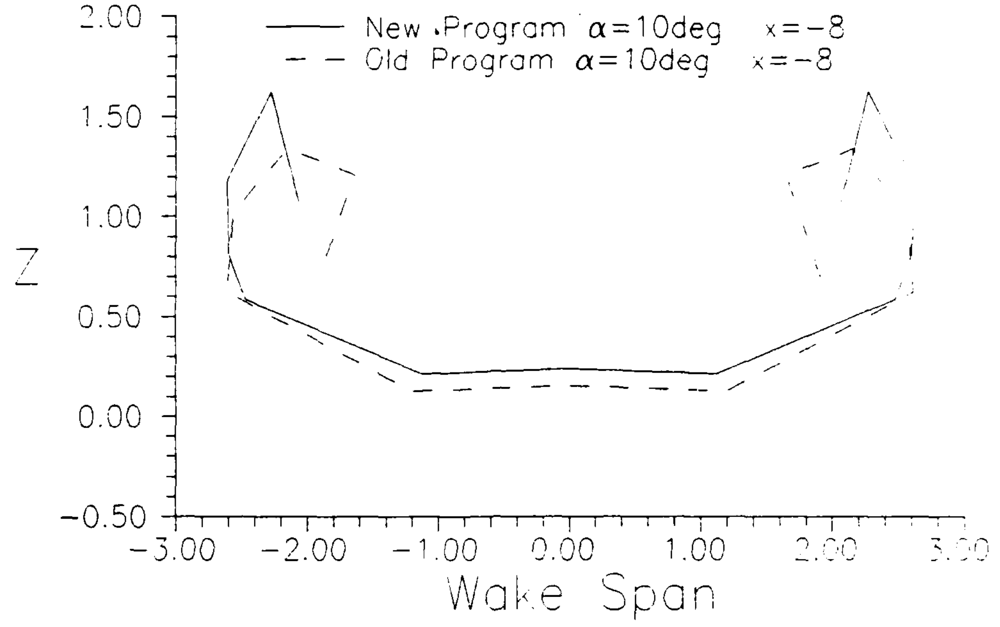

Figure 16c Wake Cross Sections (A.O.A. = 10 deg., x = -8)

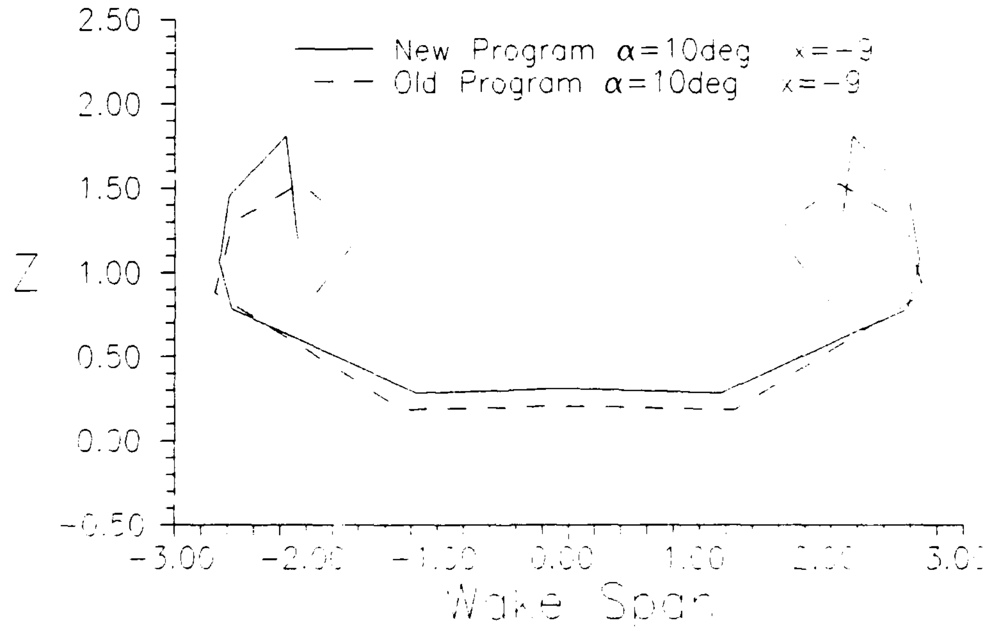

Figure 16d Wake Cross Sections (A.O.A. = 10 deg., x = -9)

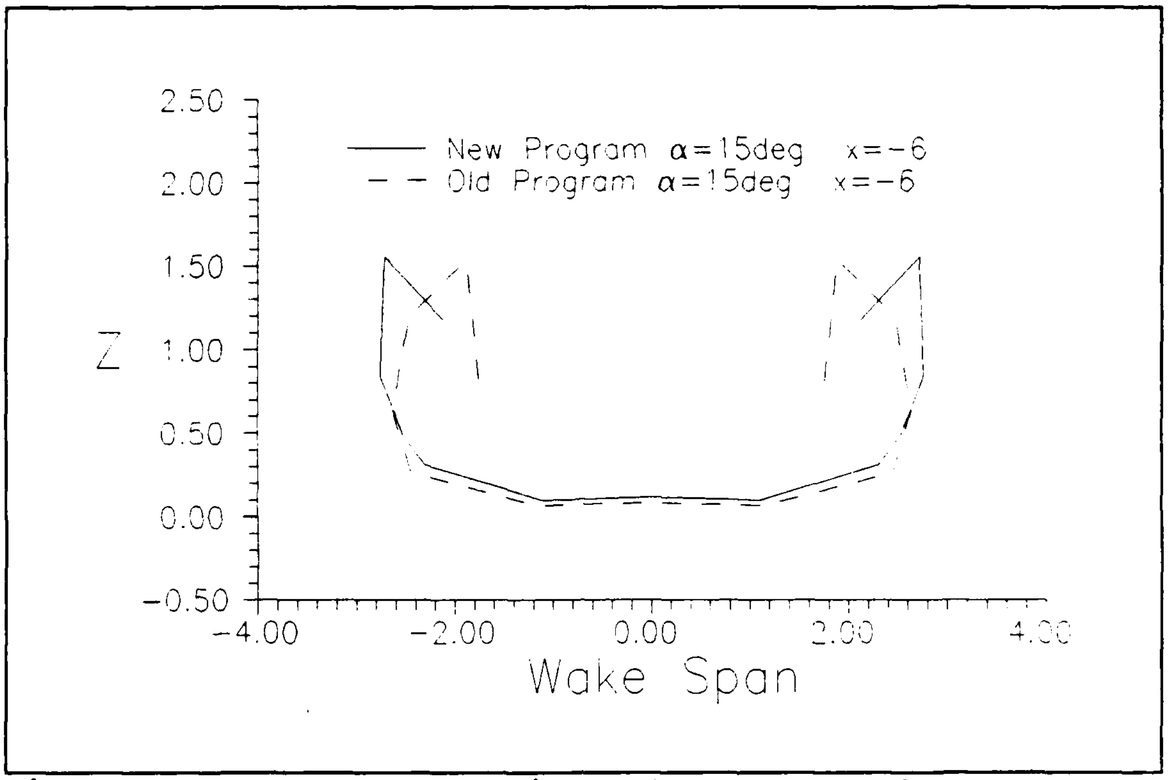

Figure 17a Wake Cross Sections (A.O.A. = 15 deg., x = -6)

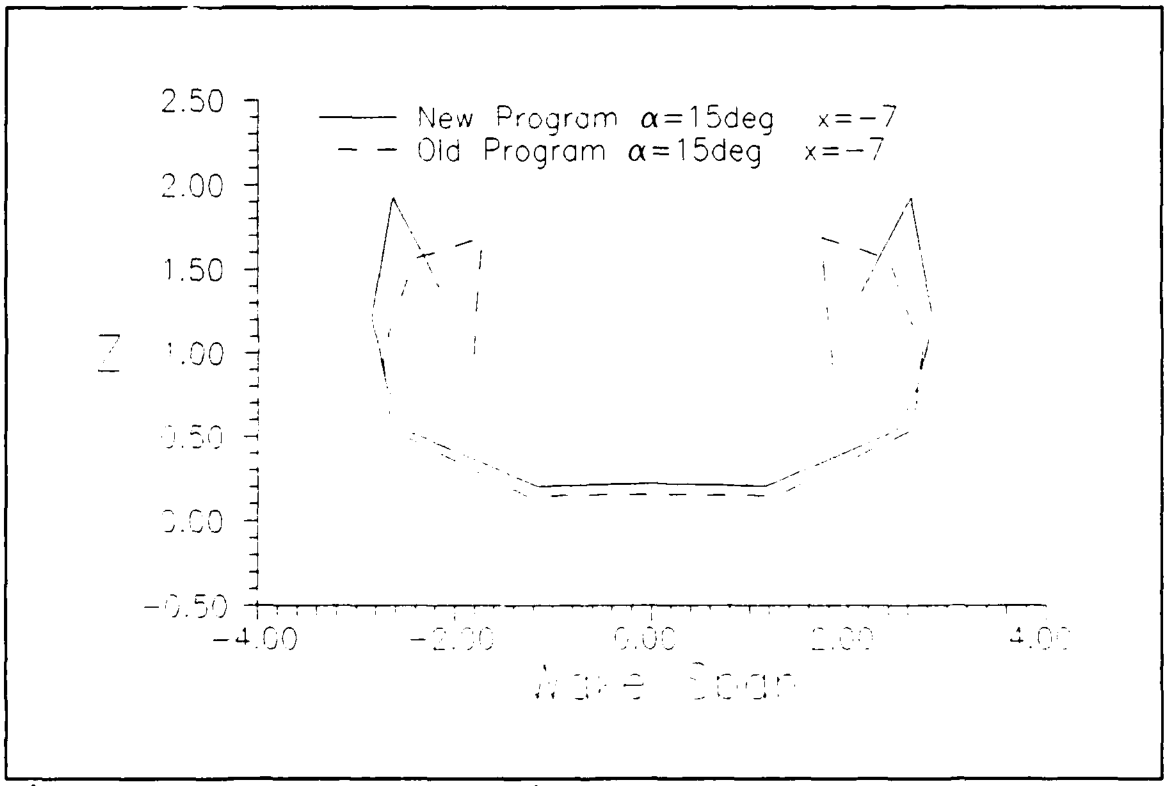

Figure 17b Wake Cross Sections (A.O.A. = 15 deg., x = -7)

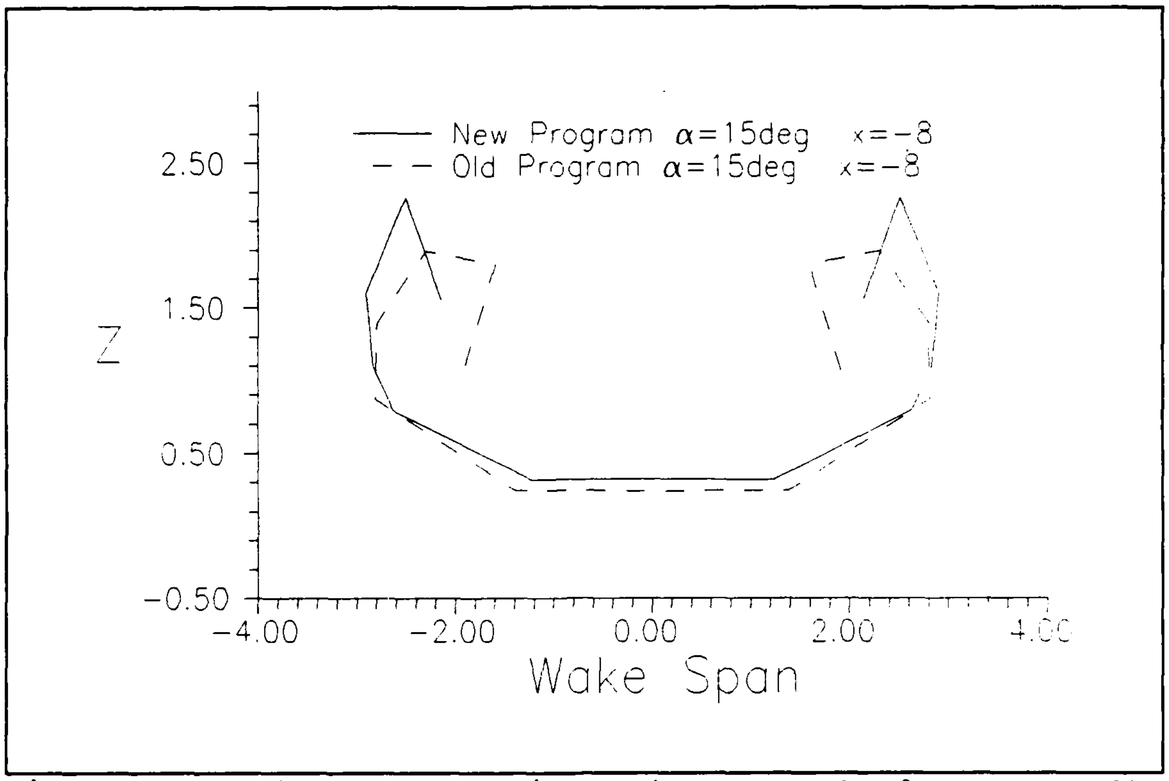

Figure 17c Wake Cross Sections (A.O.A. = 15 deg., x = -8)

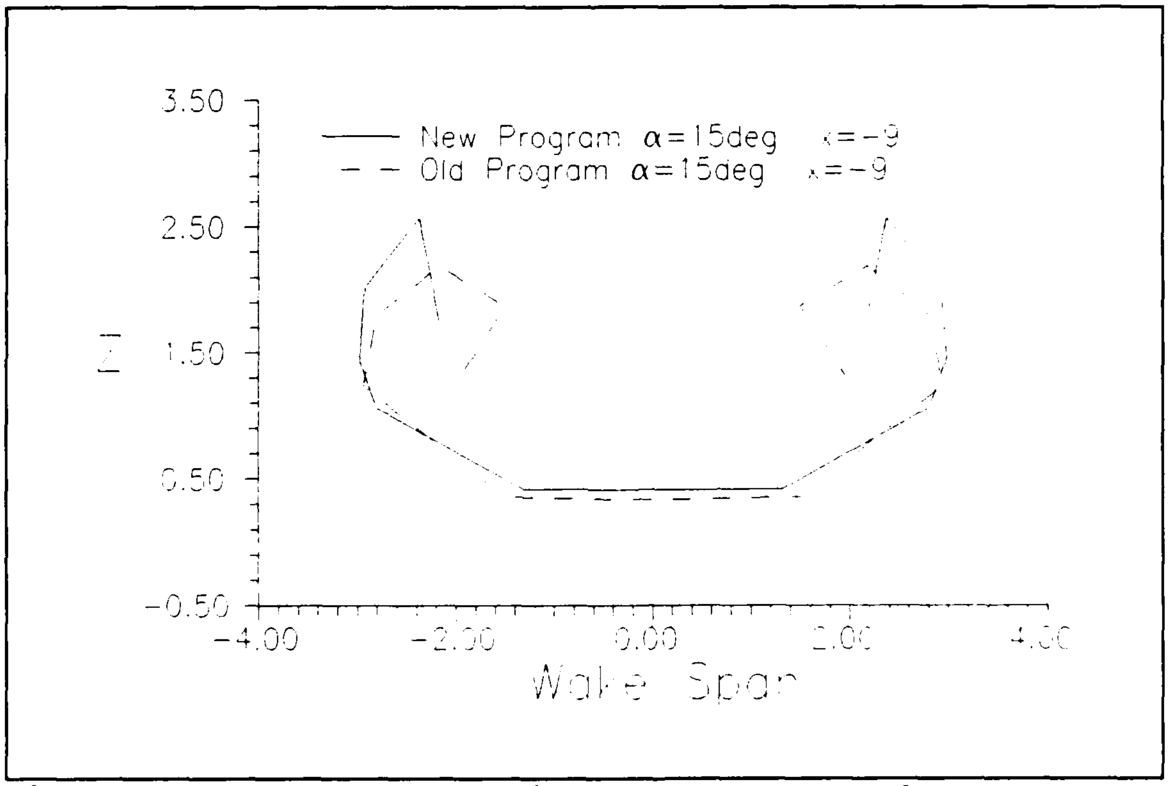

Figure 17d Wake Cross Sections (A.O.A. = 15 deg., x = -9)

V. Conclusions and Recommendations

Conclusions

An algorithm was developed to model the wake as a continuous vortex sheet in a potential-flow solution which had modeled the wake as a lattice of discrete vortex cores. Although the program met with numerical difficulties, results obtained from limited simulations seem to indicate that the theory and method are, in general, valid and worthy of further pursuit. Two areas are suspect as being responsible for the difficulties encountered at high angles of attack and for the delta wing.

The first possible shortcoming is that the present effort's program does not have the capability to "split" vortex cores in the wake as they grow beyond a pre-established length. The original program has this capability which smooths the approximation of the wake, redistributes the vorticities, and allows for tighter rolling up of the wake sheet. The exclusion of this capability in the present work was a conscious decision for the sake of simplifying the discretization scheme until the theory is validated. The capability could be re-established by revising the indexing system for the elements and nodes in the wake. This implies a more difficult bookkeeping task, but not an adjustment in the basic theory.

The second, and probably most critical area is the method of assigning the vorticity at the newly defined midpoints of the quadrilateral wake elements. By simply assigning the average of the four corners, the divergence condition of Equation (21) is not satisfied at these points. The erroneous vorticities result in erroneous induced velocities. These in turn distort the positions and orientations of vorticities for the next time step, and a snowball effect ensues. This difficulty manifested itself more severely in high angle of attack and delta wing configurations where interaction of the vortex cores is more pronounced, and unrealistically high induced velocities were calculated.

Both of these shortcomings may be contributing in varying degrees to the error in the simulation, but both *are reparable*. It is *expected* that their correction would eliminate the numerical problems.

Recommendations

There are four recommended areas for further study in this research. The first two are, not surprisingly, concerned with the two problem areas discussed above. The most critical to making this a viable tool is believed to be satisfying the divergence condition at the quadrilateral midpoints. Reintroducing the wake-splitting capability would further refine the simulation, once the program is operational.

The third area, mentioned in Chapter 1, is the study of "how close is too close" for the discrete core method to be employed in calculating induced velocities. This effort, too, is secondary to resolving the numerical difficulties, but would greatly decrease the computer time required for any simulations. This would require developing a point-to-plane distance calculation to determine whether the point of interest is within a prescribed cutoff distance of the vortex sheet.

The fourth area concerns the fact that this program is limited to symmetric wings. The older version of the program had this limitation due to the method of assigning updated position coordinates to wake nodes for the next time step. A similar technique for mirroring across midspan was used in this work for assigning vorticities to wake nodes, further entrenching the program into the symmetric requirement. Minor modifications would be required to remove this restriction, making it a more powerful design tool.

Appendix A: Subroutine CONVE

Subroutine CONVE convects the wake by calculating the total velocity at each wake position and determining the new position for the next time step using the first order finite difference formula. The G_i coefficients corrected for expansion and deformation of the wake element are calculated for both the 1- and 2-positions and passed along with the average circulation value for the quadrilateral wake element. To calculate the total particle velocity, CONVE calls subroutine VELLS (velocity due to freestream) and VELBND (velocity due to the bound vorticity on the wing) already in place in the existing program. CONVE also calls VELWK for contribution of the wake (Appendix C).

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                              C
C                        *************                         C
C                        *   CONVE   *                         C
C                        *************                         C
C                                                              C
C    THIS SUBROUTINE CONVECTS THE WAKE AT THE LOCAL VELOCITY.  C
C    THE Gi COEFF.'s OF THE G-AVERAGE EQUATION (CORRECTED FOR  C
C    ELEMENT GROWTH) ARE CALCULATED AND CONVECTED ALONG WITH   C
C    G-AVERAGES AND WAKE POSITION COORDINATES.  THIS ROUTINE   C
C    CALLS SUBROUTINE VELBND TO DETERMINE VELOCITY INDUCED BY  C
C    THE BODY AND VELWK TO DETERMINE VELOCITY INDUCED BY THE   C
C    WAKE WHICH HAS BEEN MODELED AS A CONTINUOUS VORTEX SHEET. C
C                                                              C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE CONVE
      IMPLICIT REAL*8(A-H,O-Z)
      PARAMETER (ME=1000,MN=1000,MEQ=200,MUN=200,MC=25,MCI=100)
      PARAMETER (MW=25,MWP=25,MGP=240,IDU=7,M1=3,M2=3)
      PARAMETER (MWP1=101)
      DIMENSION AGP(MGP,7),AGPT(7,MGP),BGP(MGP),MSH(MGP),GP(MGP)
      COMMON/VBOUND/ VBX,VBY,VBZ
      COMMON/VLSURF/ VLSX,VLSY,VLSZ
      COMMON/VEWAKE/ VWKX,VWKY,VWKZ
      COMMON/PO1WAK/ X1WAK(MC,MCI,MW),[illegible]AK(MC,MCI,MW),Z1WAK(MC,MCI,MW)
      COMMON/PO2WAK/ X2WAK(MC,MCI,MW),Y2WAK(MC,MCI,MW),Z2WAK(MC,MCI,MW)
      COMMON/STRWAK/ GAVE(MC,MCI),GWAKE(MC,MCI,MW),NRWAKE(MC,MW)
      COMMON/GNUMBE/ G1(MC,MCI),G2(MC,MCI),G3(MC,MCI)
      COMMON/WKCOEF/ G1WAKE(MC,MCI,MW),G2WAKE(MC,MCI,MW),
     .               G3WAKE(MC,MCI,MW),G1BAR(MC,MCI,MW),
     .               G2BAR(MC,MCI,MW),G3BAR(MC,MCI,MW)
      COMMON/GBYEDG/ G1W(MC,MCI),G2W(MC,MCI),G3W(MC,MCI)
      COMMON/WKDIST/ DPDD(MC,MCI,MW),D(MC,MCI,MW)
      COMMON/SPLITW/ ISPLIT(MC,MCI,MW),DIMINI,ISPER(MCI,MW)
      COMMON/SPCOUN/ IAT1(MW),IAT2(MW),LAST1(MW),LAST2(MW)
      COMMON/SPCOU1/ NCOUNT(MW),GCON(MCI)
      COMMON/SP1WAK/ X1PER(MCI,MW),Y1PER(MCI,MW),Z1PER(MCI,MW)
      COMMON/SP2WAK/ X2PER(MCI,MW),Y2PER(MCI,MW),Z2PER(MCI,MW)
      COMMON/SPGWAK/ ISPL(MCI,MW),G(MCI,MW),GPER(MCI,MW)
      COMMON/SPWOR1/ X1(MCI,MW),Y1(MCI,MW),Z1(MCI,MW)
      COMMON/SPWOR2/ X2(MCI,MW),Y2(MCI,MW),Z2(MCI,MW)
      COMMON/EDDATA/ NEDGE,NED(MCI,2),NCONV,NCO(MCI,3),NWKP,NWAKE1(MWP)
      COMMON/EDGEDA/ NOPEN,NCLOSE,NTOTCR,NEDG(MC,MCI,2),NCIRC(MC)
      COMMON/CONVEC/ NCOVCR,NCONCR(MCI),NCON(MC,MCI,2),NCONPO(MCI)
      COMMON/CONWOR/ XNP(MC,MCI,MWP1),YNP(MC,MCI,MWP1),ZNP(MC,MCI,MWP1)
      COMMON/SP3WAK/ X2TEM(MC,MCI,MW),Y2TEM(MC,MCI,MW),Z2TEM(MC,MCI,MW)
      COMMON/VELINP/ XP,YP,ZP,IDENTE,NODE1,NODE2
      COMMON/TIMESD/ TIME,DTIME,NMOTIO,NCURTM,NTIME,NSTIME,NWPTS,MAXWK
      COMMON/TIMESE/ NSTMOT
      COMMON/MODATA/ PMOMX,PMOMY,PMOMZ,SPAN,CHORD
      COMMON/ERRORS/ IERROR,KERROR,MERROR,NERROR
      COMMON/CNTERS/ KCIR,NWELE

      OFF=0.03D0*CHORD
C
C       DOING ONE CIRCUIT AT A TIME
C
      DO 900 KCIR=1,NCOVCR
C
C       INITIALIZING THE SPLITTING OF A NEW ROW
C
        DO 20 I=1,NRWAKE(KCIR,NWPTS+1)-1
          ISPLIT(KCIR,I,NWPTS+1)=0
   20   CONTINUE
C
C       FINDING THE CONVECTED WAKE POSITIONS
C
        DO 50 I=1,NWPTS+1
C
CCCCCCCC THIS IS FOR A SYMMETRIC WING
C
          DO 30 J=1,NRWAKE(KCIR,I)/2+1
               XP=X1WAK(KCIR,J,I)
               YP=Y1WAK(KCIR,J,I)
               ZP=Z1WAK(KCIR,J,I)
C
C       FINDING THE TOTAL VELOCITY
C
                 CALL VELBND
```

```
                  CALL VELWK
                  CALL VELLS
                  VABSX=VBX+VWKX-VLSX
                  VABSY=VBY+VWKY-VLSY
                  VABSZ=VBZ+VWKZ-VLSZ
C
C        SAVING THE NEW POSITIONS
C
                 XNP(KCIR,J,I)=XP+VABSX*DTIME
                 YNP(KCIR,J,I)=YP+VABSY*DTIME
                 ZNP(KCIR,J,I)=ZP+VABSZ*DTIME
C
C        SEEING IF THE POINT IS TOO CLOSE TO THE WING
C
                 CALL ISITOV(XNP(KCIR,J,I),YNP(KCIR,J,I),ZNP(KCIR,J,I)
     .                            ,OFF)
   30          CONTINUE
               DO 40 J=1,NRWAKE(KCIR,I)/2
                   K=NRWAKE(KCIR,I)-J+1
                   XNP(KCIR,K,I)=XNP(KCIR,J,I)
                   YNP(KCIR,K,I)=-YNP(KCIR,J,I)
                   ZNP(KCIR,K,I)=ZNP(KCIR,J,I)
   40          CONTINUE
   50    CONTINUE
*********************************************
      IF(NWPTS.NE.0)THEN
*********************************************
C
C        MOVING THE BAR (1-POSITION) CIRCULATION BACK TO FRONT
C
      ICO=0
      DO 320 II=1,NWPTS
           I=NWPTS-ICO
           IP1=I+1
           NSQ=NRWAKE(KCIR,I)-1
           NEQ=3*NSQ
c
c  Create "mshift" array for "packing" the banded matrix
c  used to solve for G-primes.
c
         DO 280 NK=1,NEQ
           MSH(NK)=0
               IF(NK .GT. 4) MSH(NK)=4-NK
 280     CONTINUE
c
c  Zeroing out the matrix equation for G-primes.
c
      DO 285 ICL=1,NEQ
        BGP(ICL)=0.0D0
        DO 285 JCL=1,7
          AGP(ICL,JCL)=0.0D0
 285  CONTINUE
```

```
c
c  Building the G-prime matrix equation "packed".  The equation
c  is banded to 3 elements either side of the main diagonal.  The
c  packing offsets the rows by 'mshift' elements to form a
c  NEQ by 7 matrix.
c
c  Load row one of G-prime matrix eq.-- the left end condition.
c
          AGP(1,3)=1.0D0
          BGP(1)=G3BAR(KCIR,1,I)
      DO 290 J=1,NSQ
            JP1=J+1
c
c  Calculate d squared, (d + delta d) squared for G-prime Eq.'s.
c
            DSQ=((X1WAK(KCIR,JP1,I)-X1WAK(KCIR,J,I))
     .          *(X1WAK(KCIR,JP1,I)-X1WAK(KCIR,J,I)))+
     .          ((Y1WAK(KCIR,JP1,I)-Y1WAK(KCIR,J,I))
     .          *(Y1WAK(KCIR,JP1,I)-Y1WAK(KCIR,J,I)))+
     .          ((Z1WAK(KCIR,JP1,I)-Z1WAK(KCIR,J,I))
     .          *(Z1WAK(KCIR,JP1,I)-Z1WAK(KCIR,J,I)))
            DPDDSQ=((XNP(KCIR,JP1,I)-XNP(KCIR,J,I))
     .             *(XNP(KCIR,JP1,I)-XNP(KCIR,J,I)))+
     .             ((YNP(KCIR,JP1,I)-YNP(KCIR,J,I))
     .             *(YNP(KCIR,JP1,I)-YNP(KCIR,J,I)))+
     .             ((ZNP(KCIR,JP1,I)-ZNP(KCIR,J,I))
     .             *(ZNP(KCIR,JP1,I)-ZNP(KCIR,J,I)))
            DPDD(KCIR,J,IP1)=DSQRT(DPDDSQ)
            D(KCIR,J,IP1)=DSQRT(DSQ)
c
c  Indexing to build banded format
c
            IN1=3*J-1
            IN2=3*J-2
c
c  Load G-average Eq. into matrix
c
            AGP(IN1,IN2+MSH(IN1))=DPDDSQ/6.0D0
            AGP(IN1,IN2+1+MSH(IN1))=DPDD(KCIR,J,IP1)/2.0D0
            AGP(IN1,IN2+2+MSH(IN1))=1.0D0
            BGP(IN1)=-GWAKE(KCIR,J,I)

           IF (J .LT. NSQ) THEN
c
c  Load Magnitude Compatibility Equation
c
              IN1=IN1+1
              AGP(IN1,IN2+MSH(IN1))=DPDDSQ/2.0D0
              AGP(IN1,IN2+1+MSH(IN1))=DPDD(KCIR,J,IP1)
              AGP(IN1,IN2+2+MSH(IN1))=1.0D0
              AGP(IN1,IN2+5+MSH(IN1))=-1.0D0
```

```
c
c  Load Slope Compatibility Eq.
c
                    IN1=IN1+1
                    AGP(IN1,IN2+MSH(IN1))=DPDD(KCIR,J,IP1)
                    AGP(IN1,IN2+1+MSH(IN1))=1.0D0
                    AGP(IN1,IN2+4+MSH(IN1))=-1.0D0
c
c  If last element, load right end condition.
c
                 ELSE
                    IN1=IN1+1
                    AGP(IN1,IN2+MSH(IN1))=DPDDSQ/2.0D0
                    AGP(IN1,IN2+1+MSH(IN1))=DPDD(KCIR,J,IP1)
                    AGP(IN1,IN2+2+MSH(IN1))=1.0D0
                    BGP(IN1)=G1BAR(KCIR,J,I)*DSQ/2.0D0+G2BAR(KCIR,J,I)
     .                      *D(KCIR,J,IP1)+G3BAR(KCIR,J,I)
                 END IF

 290           CONTINUE
c
c  Transpose AGP for solver
c

               DO 300 IT=1,NEQ
                 DO 300 JT=1,7
                   AGPT(JT,IT)=AGP(IT,JT)
 300             CONTINUE

c                                                T
c  Call Banded matrix solver for [AGP] [GP]=[BGP]
c
               CALL MSOLV(AGPT,GP,BGP,M1,M2,IDU,NEQ)
c
c  Assign Position-1 coefficients in GBAR arrays.
c
              DO 310 J=1,NRWAKE(KCIR,I)-1
               IND=3*J-2
               G1BAR(KCIR,J,IP1)=GP(IND)
               G2BAR(KCIR,J,IP1)=GP(IND+1)
               G3BAR(KCIR,J,IP1)=GP(IND+2)
 310         CONTINUE
               ICO=ICO+1
 320     CONTINUE
C
C          MOVING THE CIRCULATION BACK TO FRONT
C
         ICO=0
         DO 120 II=1,NWPTS
              I=NWPTS-ICO
              IP1=I+1
              NSQ=NRWAKE(KCIR,I)-1
              NEQ=3*NSQ
```

```
c
c  Create "mshift" array for "packing" the banded matrix
c  used to solve for G-primes.
c
            DO 80 NK=1,NEQ
              MSH(NK)=0
                  IF(NK .GT. 4) MSH(NK)=4-NK
  80        CONTINUE
c
c  Zeroing out the matrix equation for G-primes.
c
        DO 85 ICL=1,NEQ
          BGP(ICL)=0.0D0
          DO 85 JCL=1,7
            AGP(ICL,JCL)=0.0D0
  85    CONTINUE
c
c  Building the G-prime matrix equation "packed".  The equation
c  is banded to 3 elements either side of the main diagonal.  The
c  packing offsets the rows by 'mshift' elements to form a
c  NEQ by 7 matrix.
c
c  Load row one of G-prime matrix eq.-- the left end condition.
c
              AGP(1,3)=1.0D0
              BGP(1)=G3WAKE(KCIR,1,I)

              DO 90 J=1,NSQ
                JP1=J+1
c
c  Calculate d squared, (d + delta d) squared for G-prime Eq.'s.
c
                DSQ=((X1WAK(KCIR,JP1,IP1)-X1WAK(KCIR,J,IP1))
     .                *(X1WAK(KCIR,JP1,IP1)-X1WAK(KCIR,J,IP1)))+
     .                ((Y1WAK(KCIR,JP1,IP1)-Y1WAK(KCIR,J,IP1))
     .                *(Y1WAK(KCIR,JP1,IP1)-Y1WAK(KCIR,J,IP1)))+
     .                ((Z1WAK(KCIR,JP1,IP1)-Z1WAK(KCIR,J,IP1))
     .                *(Z1WAK(KCIR,JP1,IP1)-Z1WAK(KCIR,J,IP1)))
                DPDDSQ=((XNP(KCIR,JP1,IP1)-XNP(KCIR,J,IP1))
     .                   *(XNP(KCIR,JP1,IP1)-XNP(KCIR,J,IP1)))+
     .                   ((YNP(KCIR,JP1,IP1)-YNP(KCIR,J,IP1))
     .                   *(YNP(KCIR,JP1,IP1)-YNP(KCIR,J,IP1)))+
     .                   ((ZNP(KCIR,JP1,IP1)-ZNP(KCIR,J,IP1))
     .                   *(ZNP(KCIR,JP1,IP1)-ZNP(KCIR,J,IP1)))
                DPDD(KCIR,J,IP1)=DSQRT(DPDDSQ)
                D(KCIR,J,IP1)=DSQRT(DSQ)
c
c  Indexing to build banded format
c
                IN1=3*J-1
                IN2=3*J-2
```

```
c
c  Load G-average Eq. into matrix
c
                AGP(IN1,IN2+MSH(IN1))=DPDDSQ/6.0D0
                AGP(IN1,IN2+1+MSH(IN1))=DPDD(KCIR,J,IP1)/2.0D0
                AGP(IN1,IN2+2+MSH(IN1))=1.0D0
                BGP(IN1)=GWAKE(KCIR,J,I)

               IF (J .LT. NSQ) THEN
c
c  Load Magnitude Compatibility Equation
c
                   IN1=IN1+1
                   AGP(IN1,IN2+MSH(IN1))=DPDDSQ/2.0D0
                   AGP(IN1,IN2+1+MSH(IN1))=DPDD(KCIR,J,IP1)
                   AGP(IN1,IN2+2+MSH(IN1))=1.0D0
                   AGP(IN1,IN2+5+MSH(IN1))=-1.0D0
c
c  Load Slope Compatibility Eq.
c
                   IN1=IN1+1
                   AGP(IN1,IN2+MSH(IN1))=DPDD(KCIR,J,IP1)
                   AGP(IN1,IN2+1+MSH(IN1))=1.0D0
                   AGP(IN1,IN2+4+MSH(IN1))=-1.0D0
c
c  If last element, load right end condition.
c
                ELSE
                   IN1=IN1+1
                   AGP(IN1,IN2+MSH(IN1))=DPDDSQ/2.0D0
                   AGP(IN1,IN2+1+MSH(IN1))=DPDD(KCIR,J,IP1)
                   AGP(IN1,IN2+2+MSH(IN1))=1.0D0
                   BGP(IN1)=G1WAKE(KCIR,J,I)*DSQ/2.0D0+G2WAKE(KCIR,J,I)
         .                 *D(KCIR,J,IP1)+G3WAKE(KCIR,J,I)
                END IF

  90          CONTINUE
c
c  Transpose AGP for solver
c

              DO 100 IT=1,NEQ
                DO 100 JT=1,7
                  AGPT(JT,IT)=AGP(IT,JT)
 100            CONTINUE

c                                         T
c  Call Banded matrix solver for [AGP] [GP]=[BGP]
c
              CALL MSOLV(AGPT,GP,BGP,M1,M2,IDU,NEQ)
```

```
c
c  Assign G-prime answers (Position-2 coefficients)
c  to appropriate G#WAKE arrays.
c
         DO 110 J=1,NRWAKE(KCIR,I)-1
           IND=3*J-2
           G1WAKE(KCIR,J,IP1)=GP(IND)
           G2WAKE(KCIR,J,IP1)=GP(IND+1)
           G3WAKE(KCIR,J,IP1)=GP(IND+2)
           GWAKE(KCIR,J,IP1)=GWAKE(KCIR,J,I)
 110     CONTINUE
           ICO=ICO+1
 120  CONTINUE
      END IF
C
C       UPDATING THE POSITIONS FIRST THE X1 POSITIONS
C
      DO 70 I=1,NWPTS+1
           IP1=I+1
           DO 60 J=1,NRWAKE(KCIR,I)
                X1WAK(KCIR,J,IP1)=XNP(KCIR,J,I)
                Y1WAK(KCIR,J,IP1)=YNP(KCIR,J,I)
                Z1WAK(KCIR,J,IP1)=ZNP(KCIR,J,I)
                X2WAK(KCIR,J,I)=XNP(KCIR,J,I)
                Y2WAK(KCIR,J,I)=YNP(KCIR,J,I)
                Z2WAK(KCIR,J,I)=ZNP(KCIR,J,I)
   60           CONTINUE
   70 CONTINUE

      NRWAKE(KCIR,NWPTS+2)=NRWAKE(KCIR,NWPTS+1)
*****************************************************************
C       MOVING THE WAKE STRENGTH OFF THE WING (G-prime corrected
C       for expansion.)
*****************************************************************
       NEQ=3*NCONCR(KCIR)
c
c  Create "mshift" array for "packing" the banded matrix
c  used to solve for G-primes.
c
         DO 130 NK=1,NEQ
           MSH(NK)=0
           IF(NK .GT. 4) MSH(NK)=4-NK
 130     CONTINUE
c
c  Zeroing out the matrix equation for G-primes.
c
      DO 140 ICL=1,NEQ
        BGP(ICL)=0.0D0
        DO 140 JCL=1,7
          AGP(ICL,JCL)=0.0D0
 140  CONTINUE
```

```
c
c  Building the G-prime matrix equation "packed".  The equation
c  is banded to 3 elements either side of the main diagonal.  The
c  packing offsets the rows by 'mshift' elements to form a
c  NEQ by 7 matrix.
c
c  Load row one of G-prime matrix eq.-- the left end condition.
c
      AGP(1,3)=1.0D0
      BGP(1)=G3(NCON(KCIR,1,1),NCON(KCIR,1,2))

      DO 150 J=1,NCONCR(KCIR)
         JP1=J+1
         L=NCON(KCIR,J,1)
         M=NCON(KCIR,J,2)
c
c  Calculate d squared, (d + delta d) squared for G-prime Eq.'s.
c
         DSQ=((X1WAK(KCIR,JP1,1)-X1WAK(KCIR,J,1))
     .           *(X1WAK(KCIR,JP1,1)-X1WAK(KCIR,J,1)))+
     .           ((Y1WAK(KCIR,JP1,1)-Y1WAK(KCIR,J,1))
     .           *(Y1WAK(KCIR,JP1,1)-Y1WAK(KCIR,J,1)))+
     .           ((Z1WAK(KCIR,JP1,1)-Z1WAK(KCIR,J,1))
     .           *(Z1WAK(KCIR,JP1,1)-Z1WAK(KCIR,J,1)))
         DPDDSQ=((XNP(KCIR,JP1,1)-XNP(KCIR,J,1))
     .              *(XNP(KCIR,JP1,1)-XNP(KCIR,J,1)))+
     .              ((YNP(KCIR,JP1,1)-YNP(KCIR,J,1))
     .              *(YNP(KCIR,JP1,1)-YNP(KCIR,J,1)))+
     .              ((ZNP(KCIR,JP1,1)-ZNP(KCIR,J,1))
     .              *(ZNP(KCIR,JP1,1)-ZNP(KCIR,J,1)))
         DPDD(KCIR,J,1)=DSQRT(DPDDSQ)
         D(KCIR,J,1)=DSQRT(DSQ)
c
c  Indexing to build banded format
c
         IN1=3*J-1
         IN2=3*J-2
c
c  Load G-average Eq. into matrix
c
         AGP(IN1,IN2+MSH(IN1))=DPDDSQ/6.0D0
         AGP(IN1,IN2+1+MSH(IN1))=DPDD(KCIR,J,1)/2.0D0
         AGP(IN1,IN2+2+MSH(IN1))=1.0D0
         BGP(IN1)=GAVE(L,M)
       IF (J .LT. NCONCR(KCIR)) THEN
c
c  Load Magnitude Compatibility Equation
c
        IN1=IN1+1
           AGP(IN1,IN2+MSH(IN1))=DPDDSQ/2.0D0
           AGP(IN1,IN2+1+MSH(IN1))=DPDD(KCIR,J,1)
           AGP(IN1,IN2+2+MSH(IN1))=1.0D0
           AGP(IN1,IN2+5+MSH(IN1))=-1.0D0
```

```
c
c  Load Slope Compatibility Eq.
c
                IN1=IN1+1
                AGP(IN1,IN2+MSH(IN1))=DPDD(KCIR,J,1)
                AGP(IN1,IN2+1+MSH(IN1))=1.0D0
                AGP(IN1,IN2+4+MSH(IN1))=-1.0D0
c
c  If last element, load right end condition.
c
           ELSE
                IN1=IN1+1
                AGP(IN1,IN2+MSH(IN1))=DPDDSQ/2.0D0
                AGP(IN1,IN2+1+MSH(IN1))=DPDD(KCIR,J,1)
                AGP(IN1,IN2+2+MSH(IN1))=1.0D0
                BGP(IN1)=(G1(L,M)*DSQ/2.0D0)+(G2(L,M)*D(KCIR,J,1))+G3(L,M)
           END IF

 150  CONTINUE
c
c  Transpose AGP for solver
c
      DO 160 IT=1,NEQ
        DO 160 JT=1,7
          AGPT(JT,IT)=AGP(IT,JT)
 160  CONTINUE
c                                                T
c  Call Banded matrix solver for [AGP] [GP]=[BGP]
c
      CALL MSOLV(AGPT,GP,BGP,M1,M2,IDU,NEQ)
c
c  Assign Position-1 coefficients in GBAR arrays and G-prime answers
c  (Position-2 coefficients) to appropriate G#WAKE arrays.
c
      DO 170 J=1,NRWAKE(KCIR,I)-1
         L=NCON(KCIR,J,1)
         M=NCON(KCIR,J,2)
         G1BAR(KCIR,J,1)=-G1(L,M)
         G2BAR(KCIR,J,1)=-G2(L,M)
         G3BAR(KCIR,J,1)=-G3(L,M)
         IND=3*J-2
         G1WAKE(KCIR,J,1)=GP(IND)
         G2WAKE(KCIR,J,1)=GP(IND+1)
         G3WAKE(KCIR,J,1)=GP(IND+2)
         GWAKE(KCIR,J,1)=GAVE(L,M)
 170  CONTINUE

 900  CONTINUE

      RETURN
      END
```

Appendix B: Subroutine CONSHT

Subroutine CONSHT builds the wake as a continuous vortex sheet by appending wake information onto the existing arrays that define the body (wing) as a bound vortex sheet. These values are required by subroutines VELWK (Appendix C), and VELE and VELVF of the existing program to calculate the disturbance velocity induced by the wake when modeled as a collection of triangular elements of linearly varying vorticity.

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                C
C                       ****************                         C
C                       *    CONSHT    *                         C
C                       ****************                         C
C                                                                C
C  THIS SUBROUTINE BUILDS THE CONTINUOUS WAKE SHEET BY APPEND-    C
C  ING WAKE INFORMATION ONTO ARRAYS THAT DEFINE THE BODY.         C
C                                                                C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE CONSHT
      IMPLICIT REAL*8(A-H,O-Z)
      PARAMETER (ME=1000,MN=1000,MEQ=200,MUN=200,MC=25,MCI=100)
      PARAMETER (MW=25,MWP=25,MGP=240)
      COMMON/WKCOEF/ G1WAKE(MC,MCI,MW),G2WAKE(MC,MCI,MW),
     .               G3WAKE(MC,MCI,MW),G1BAR(MC,MCI,MW),
     .               G2BAR(MC,MCI,MW),G3BAR(MC,MCI,MW)
      COMMON/GBYEDG/ G1W(MC,MCI),G2W(MC,MCI),G3W(MC,MCI)
      COMMON/ELDATA/ NELE,NODE,N(ME,3),XNODE(MN),YNODE(MN),ZNODE(MN)
      COMMON/EDDATA/ NEDGE,NED(MCI,2),NCONV,NCO(MCI,3),NWKP,NWAKE1(MWP)
      COMMON/EDGEDA/ NOPEN,NCLOSE,NTOTCR,NEDG(MC,MCI,2),NCIRC(MC)
      COMMON/PO1WAK/ X1WAK(MC,MCI,MW),Y1WAK(MC,MCI,MW),Z1WAK(MC,MCI,MW)
      COMMON/PO2WAK/ X2WAK(MC,MCI,MW),Y2WAK(MC,MCI,MW),Z2WAK(MC,MCI,MW)
      COMMON/TIMESD/ TIME,DTIME,NMOTIO,NCURTM,NTIME,NSTIME,NWPTS,MAXWK
      COMMON/CONSTS/ FOURPI,PI,CUTOFF,CUTWK
      COMMON/CONVEC/ NCOVCR,NCONCR(MCI),NCON(MC,MCI,2),NCONPO(MCI)
      COMMON/GNORMS/ ANORX(MN),ANORY(MN),ANORZ(MN),CFACT(ME,3)
      COMMON/STRWAK/ GAVE(MC,MCI),GWAKE(MC,MCI,MW),NRWAKE(MC,MW)
      COMMON/GOMEGB/ OMEX(MN),OMEY(MN),OMEZ(MN)
      COMMON/GNUMBE/ G1(MC,MCI),G2(MC,MCI),G3(MC,MCI)
      COMMON/CORNER/ AELE(ME),BELE(ME),CELE(ME),DIRCOS(ME,3,3)
      COMMON/EBASIS/ ABASE(ME,6)
      COMMON/CONPOS/ XCP(ME),YCP(ME),ZCP(ME)
      COMMON/WKDIST/ DPDD(MC,MCI,MW),D(MC,MCI,MW)
      COMMON/CNTERS/ KCIR,NWELE
```

```
**************************************************
C   Appending the x-, y-, and z-location of wake
C   nodes onto the XNODE, YNODE, and ZNODE arrays.
**************************************************
      NNODE=NODE
      KCIR=1
      DO 40 I=1,NWPTS
        DO 10 J=1,NRWAKE(KCIR,I)
          NNODE=NNODE+1
          XNODE(NNODE)=X1WAK(KCIR,J,I)
          YNODE(NNODE)=Y1WAK(KCIR,J,I)
          ZNODE(NNODE)=Z1WAK(KCIR,J,I)
  10    CONTINUE
C
C  Average coords. for "midpoint" of 4-sided wake element.
C
        DO 20 J=1,NRWAKE(KCIR,I)-1
          JP1=J+1
          NNODE=NNODE+1
          XNODE(NNODE)=(X1WAK(KCIR,J,I)+X2WAK(KCIR,J,I)
     .                 +X1WAK(KCIR,JP1,I)+X2WAK(KCIR,JP1,I))/4.0D0
          YNODE(NNODE)=(Y1WAK(KCIR,J,I)+Y2WAK(KCIR,J,I)
     .                 +Y1WAK(KCIR,JP1,I)+Y2WAK(KCIR,JP1,I))/4.0D0
          ZNODE(NNODE)=(Z1WAK(KCIR,J,I)+Z2WAK(KCIR,J,I)
     .                 +Z1WAK(KCIR,JP1,I)+Z2WAK(KCIR,JP1,I))/4.0D0
  20    CONTINUE
        DO 30 J=1,NRWAKE(KCIR,I)
          NNODE=NNODE+1
          XNODE(NNODE)=X2WAK(KCIR,J,I)
          YNODE(NNODE)=Y2WAK(KCIR,J,I)
          ZNODE(NNODE)=Z2WAK(KCIR,J,I)
 30     CONTINUE
 40   CONTINUE
***********************************************************
C   Appending the wake element definitions to the N(node,3)
C   array and the edge data for each wake wake row to NCIRC
C   and NEDG arrays.  Also create G1W, G2W, and G3W arrays
C   of wake G-coeff.'s subscripted by edge core # for VELBND
***********************************************************
      NROW=NWPTS
      NWELE=NELE
      DO 60 I=1,NROW
        NBOD=NTOTCR+I
        NCIRC(NBOD)=2*NRWAKE(KCIR,I)
        NCOL=NRWAKE(KCIR,I)-1
        N1=2*NCOL+1
        N2=N1+1
        DO 50 J=1,NCOL
```

```
C                                                          1 * - * 2
C       TRIANGLE #1 OF THE WAKE SQUARE                        \ /
C                                                              * 3
        NWELE=NWELE+1
        N(NWELE,1)=NODE+(I-1)*(3*NCOL+2)+J
        N(NWELE,2)=N(NWELE,1)+1
        N(NWELE,3)=NODE+(I-1)*(3*NCOL+2)+NCOL+1+J
        NEDG(NBOD,J,1)=N(NWELE,1)
        NEDG(NBOD,J,2)=N(NWELE,2)
        G1W(NBOD,J)=G1BAR(KCIR,J,I)
        G2W(NBOD,J)=G2BAR(KCIR,J,I)
        G3W(NBOD,J)=G3BAR(KCIR,J,I)
C                                                                * 1
C       TRIANGLE #2 OF THE WAKE SQUARE                      3 *  |
C                                                                * 2
        NWELE=NWELE+1
        N(NWELE,1)=N(NWELE-1,2)
        N(NWELE,2)=NODE+(I-1)*(3*NCOL+2)+2*NCOL+2+J
        N(NWELE,3)=N(NWELE-1,3)
        IF (J .EQ. NCOL) THEN
          NEDG(NBOD,N2,1)=N(NWELE,2)
          NEDG(NBOD,N2,2)=N(NWELE,1)
          G1W(NBOD,N2)=0.0D0
          G2W(NBOD,N2)=0.0D0
          G3W(NBOD,N2)=G1WAKE(KCIR,J,I)*DPDD(KCIR,J,I)*DPDD(KCIR,J,I)
     .                /2.0D0+G2WAKE(KCIR,J,I)*DPDD(KCIR,J,I)
     .                +G3WAKE(KCIR,J,I)
        END IF
C                                                              * 3
C       TRIANGLE #3 OF THE WAKE SQUARE                        / \
C                                                          2 * - * 1
        NWELE=NWELE+1
        N(NWELE,1)=N(NWELE-1,2)
        N(NWELE,2)=NODE+(I-1)*(3*NCOL+2)+2*NCOL+1+J
        N(NWELE,3)=N(NWELE-1,3)
        NEDG(NBOD,J+NCOL,1)=N(NWELE,2)
        NEDG(NBOD,J+NCOL,2)=N(NWELE,1)
        G1W(NBOD,J+NCOL)=G1WAKE(KCIR,J,I)
        G2W(NBOD,J+NCOL)=G2WAKE(KCIR,J,I)
        G3W(NBOD,J+NCOL)=G3WAKE(KCIR,J,I)
C                                                          2 *
C       TRIANGLE #4 OF THE WAKE SQUARE                       |   * 3
C                                                          1 *
        NWELE=NWELE+1
        N(NWELE,1)=N(NWELE-1,2)
        N(NWELE,2)=N(NWELE-3,1)
        N(NWELE,3)=N(NWELE-1,3)
        IF (J .EQ. 1) THEN
          NEDG(NBOD,N1,1)=N(NWELE,2)
          NEDG(NBOD,N1,2)=N(NWELE,1)
          G1W(NBOD,N1)=0.0D0
          G2W(NBOD,N1)=0.0D0
          G3W(NBOD,N1)=G3WAKE(KCIR,J,I)
        END IF
 50     CONTINUE
 60   CONTINUE
C
C       FINDING THE LONGEST SIDE FOR EACH ELEMENT
C
      DO 70 I=NELE+1,NWELE
C
C       PUTTING IN THE APPROPRIATE NODE POSITIONS INTO INTERMEDIATE
C       VARIABLES
C
          X1=XNODE(N(I,1))
          Y1=YNODE(N(I,1))
          Z1=ZNODE(N(I,1))
          X2=XNODE(N(I,2))
          Y2=YNODE(N(I,2))
          Z2=ZNODE(N(I,2))
          X3=XNODE(N(I,3))
          Y3=YNODE(N(I,3))
          Z3=ZNODE(N(I,3))
C
C       ROTATING THROUGH THE POSITIONS TO FIND THE LONGEST SIDE
C
          R1=(X2-X1)*(X2-X1)
     .      +(Y2-Y1)*(Y2-Y1)
     .      +(Z2-Z1)*(Z2-Z1)
          R2=(X3-X2)*(X3-X2)
     .      +(Y3-Y2)*(Y3-Y2)
     .      +(Z3-Z2)*(Z3-Z2)
          R3=(X1-X3)*(X1-X3)
     .      +(Y1-Y3)*(Y1-Y3)
     .      +(Z1-Z3)*(Z1-Z3)
          IF(R1.LT.R2.OR.R1.LT.R3) THEN
              IF(R2.GT.R1.AND.R2.GT.R3) THEN
C
C       FOR THE CASE WHERE THE SECOND VECTOR IS LONGEST
C
              NDUM=N(I,1)
              N(I,1)=N(I,2)
              N(I,2)=N(I,3)
              N(I,3)=NDUM
              ELSE
                  IF(R3.GT.R1.AND.R3.GT.R2)THEN
C
C       FOR THE CASE WHERE THE THIRD VECTOR IS LONGEST
C
                    NDUM=N(I,1)
                    N(I,1)=N(I,3)
                    N(I,3)=N(I,2)
                    N(I,2)=NDUM
                  END IF
               END IF
            END IF
   70    CONTINUE
*****************************************
C   Appending direction cosine matrices
C   for wake elements to DIRCOS array.
*****************************************
         DO 80 I=NELE+1,NWELE
C
C          PUTTING IN THE APPROPRIATE NODE POSITIONS INTO INTERMEDIATE
C          VARIABLES
C
            X1=XNODE(N(I,1))
            Y1=YNODE(N(I,1))
            Z1=ZNODE(N(I,1))
            X2=XNODE(N(I,2))
            Y2=YNODE(N(I,2))
            Z2=ZNODE(N(I,2))
            X3=XNODE(N(I,3))
            Y3=YNODE(N(I,3))
            Z3=ZNODE(N(I,3))
C
C          DEFINING THE INFLUENCE MATRIX
C
            D1=DSQRT((X2-X1)*(X2-X1)+(Y2-Y1)*(Y2-Y1)+(Z2-Z1)*(Z2-Z1))
            AN1=(Y2-Y1)*(Z3-Z2)-(Z2-Z1)*(Y3-Y2)
            AN2=(Z2-Z1)*(X3-X2)-(X2-X1)*(Z3-Z2)
            AN3=(X2-X1)*(Y3-Y2)-(Y2-Y1)*(X3-X2)
            D3=DSQRT(AN1*AN1+AN2*AN2+AN3*AN3)
            DIRCOS(I,1,1)=(X2-X1)/D1
            DIRCOS(I,1,2)=(Y2-Y1)/D1
            DIRCOS(I,1,3)=(Z2-Z1)/D1
            DIRCOS(I,3,1)=AN1/D3
            DIRCOS(I,3,2)=AN2/D3
            DIRCOS(I,3,3)=AN3/D3
            DIRCOS(I,2,1)=DIRCOS(I,3,2)*DIRCOS(I,1,3)
     .                   -DIRCOS(I,3,3)*DIRCOS(I,1,2)
            DIRCOS(I,2,2)=DIRCOS(I,3,3)*DIRCOS(I,1,1)
     .                   -DIRCOS(I,3,1)*DIRCOS(I,1,3)
            DIRCOS(I,2,3)=DIRCOS(I,3,1)*DIRCOS(I,1,2)
     .                   -DIRCOS(I,3,2)*DIRCOS(I,1,1)

C
C          SETTING THE CONTROL POINT LOCATION USING THE MIDPOINT OF THE
C          ELEMENT
C
            XCP(I)=(X1+X2+X3)/3.0D0
            YCP(I)=(Y1+Y2+Y3)/3.0D0
            ZCP(I)=(Z1+Z2+Z3)/3.0D0
```

```
C
C       FINDING THE VERTICIES OF THE TRIANGLE NAMELY A,B,C
C
            AELE(I)=(X2-X1)*DIRCOS(I,1,1)
     .             +(Y2-Y1)*DIRCOS(I,1,2)
     .             +(Z2-Z1)*DIRCOS(I,1,3)
            BELE(I)=(X3-X1)*DIRCOS(I,1,1)
     .             +(Y3-Y1)*DIRCOS(I,1,2)
     .             +(Z3-Z1)*DIRCOS(I,1,3)
            CELE(I)=(X3-X1)*DIRCOS(I,2,1)
     .             +(Y3-Y1)*DIRCOS(I,2,2)
     .             +(Z3-Z1)*DIRCOS(I,2,3)
C*******************************************
C  Appending the wake elements' basis
C  function coefficients to ABASE
C*******************************************
            ABASE(I,1)=-1.0D0/AELE(I)
            ABASE(I,2)=1.0D0/AELE(I)
            ABASE(I,3)=0.0D0
            ABASE(I,4)=(BELE(I)-AELE(I))/(AELE(I)*CELE(I))
            ABASE(I,5)=-BELE(I)/(AELE(I)*CELE(I))
            ABASE(I,6)=1.0D0/CELE(I)
  80  CONTINUE
C*******************************************
C  Appending the average normals for wake
C  nodes to ANORX,Y, and Z arrays.
C*******************************************
        DO 100 I=NODE+1,NNODE
            SUMX=0.0D0
            SUMY=0.0D0
            SUMZ=0.0D0
            KOUNT=0
            DO 90 J=NELE+1,NWELE
C
C       SEEING IF THE ELEMENT HAS THE NODE
C
                IF(N(J,1).EQ.I.OR.
     .             N(J,2).EQ.I.OR.
     .             N(J,3).EQ.I)THEN
C
C       ADDING IN THIS ELEMENTS NORMAL
C
                    SUMX=SUMX+DIRCOS(J,3,1)
                    SUMY=SUMY+DIRCOS(J,3,2)
                    SUMZ=SUMZ+DIRCOS(J,3,3)
                    KOUNT=KOUNT+1
                END IF
  90        CONTINUE
           ANORX(I)=SUMX/FLOAT(KOUNT)
           ANORY(I)=SUMY/FLOAT(KOUNT)
           ANORZ(I)=SUMZ/FLOAT(KOUNT)
  100 CONTINUE
```

```
C
C       MAKING UNIT NORMALS AT THE NODES
C
      DO 110 I-NODE+1,NNODE
           AMAG-DSQRT(ANORX(I)*ANORX(I)
     .                +ANORY(I)*ANORY(I)
     .                +ANORZ(I)*ANORZ(I))
           ANORX(I)-ANORX(I)/AMAG
           ANORY(I)-ANORY(I)/AMAG
           ANORZ(I)-ANORZ(I)/AMAG
  110 CONTINUE

C*****************************************
C  Appending wake node vorticities onto
C  the OMEX,Y, and Z arrays.
C*****************************************
c
c   Zero out the arrays
c
      DO 130 I-NODE+1,NNODES
        OMEX(I)-0.0D0
        OMEY(I)-0.0D0
        OMEZ(I)-0.0D0
 130  CONTINUE

      DO 150 I-1,NWPTS
        NCOL-NRWAKE(KCIR,I)-1
        DO 140 J-1,NCOL/2
C
C   The "1-Positions"
C   ****************
C   Determine the element # (triangle 1 of wake square)
C
          LEL-NELE+4*NCOL*(I-1)+4*(J-1)+1
c
c   Determine the node number for local (a,0)
c
          IND-NODE+(I-1)*(3*NCOL+2)+J
c
c  The vorticity strength at x-a.  The G-coefficint | (a,0)*-----*(0,0)
c  arrays define the strength in the negative local |       \   /
c  y-direction for triangle 1 of the squares with 0 |         *
c  at the (a,0) position.  Convert this strength to |       (b,c)
c  global coords in the positive x-direction of     |
c  triangle 4 of the square.
c
          GKEY-G2BAR(KCIR,J,I)
          OMEX(IND)-GKEY*DIRCOS(LEL+3,1,1)
          OMEY(IND)-GKEY*DIRCOS(LEL+3,1,2)
          OMEZ(IND)-GKEY*DIRCOS(LEL+3,1,3)
```

```
c
c   If at mid-span, convert the strength at local (0,0) to global
c   coords in the negative x-direction of triangle 2 of the wake
c   square.  The strength is equal to -G2bar of the next wake square.
c
          IF(J .EQ. NCOL/2) THEN
            GKEY=-G2BAR(KCIR,J+1,I)
            OMEX(IND+1)=GKEY*DIRCOS(LEL+1,1,1)
            OMEY(IND+1)=GKEY*DIRCOS(LEL+1,1,2)
            OMEZ(IND+1)=GKEY*DIRCOS(LEL+1,1,3)
            IF(MOD(NCOL,2) .EQ. 0) THEN
              OMEX(IND+1)=0.0D0
              OMEY(IND+1)=0.0D0
              OMEZ(IND+1)=0.0D0
             END IF
          END IF
C                                   |         (b,c)
C   The "2-Positions"               |           *
C   ****************                |         /   \
c                                   | (0,0) * _____ * (a,0)
c   Determine the element #         |
c   (triangle 3 of wake square)     |
c
          MEL=NELE+4*NCOL*(I-1)+4*(J-1)+3
c
c   Determine the node # for local x=0
c
          INN=NODE+(I-1)*(3*NCOL+2)+2*NCOL+1+J
c
c   Convert the strength at local (0,0) to global coords in the
c   negative x-direction of triangle 4 of the wake square.
c
          OMEX(INN)=-G2WAKE(KCIR,J,I)*DIRCOS(MEL+1,1,1)
          OMEY(INN)=-G2WAKE(KCIR,J,I)*DIRCOS(MEL+1,1,2)
          OMEZ(INN)=-G2WAKE(KCIR,J,I)*DIRCOS(MEL+1,1,3)
c
c   If at mid-span, convert the strength at local (a,0) to global
c   coords in the positive x-direction of triangle 2 of the wake square.
c   The strength is equal to G2-prime of the next wake square.
c
          IF(J .EQ. NCOL/2) THEN
            OMEX(INN+1)=-G2WAKE(KCIR,J+1,I)*DIRCOS(MEL-1,1,1)
            OMEY(INN+1)=-G2WAKE(KCIR,J+1,I)*DIRCOS(MEL-1,1,2)
            OMEZ(INN+1)=-G2WAKE(KCIR,J+1,I)*DIRCOS(MEL-1,1,3)
       IF(MOD(NCOL,2) .EQ. 0) THEN
         OMEX(INN+1)=0.0D0
         OMEY(INN+1)=0.0D0
         OMEZ(INN+1)=0.0D0
            END IF
          END IF
  140    CONTINUE
  150  CONTINUE
```

```
C
C   Append vorticities for the midpoint nodes of squares.
C                                   ********
C
      DO 180 I=1,NWPTS
        NCOL=NRWAKE(KCIR,I)-1
        DO 170 J=1,NCOL/2
c
c   Determine node # of midpoint
c
          INM=NODE+(I-1)*(3*NCOL+2)+NCOL+1+J
c
c   Determine node #'s of 4 corners around it
c
          IN1=NODE+(I-1)*(3*NCOL+2)+J
          IN2=IN1+1
          IN3=NODE+(I-1)*(3*NCOL+2)+2*NCOL+1+J
          IN4=IN3+1
c
c   Average the x-, y-, and z-components of omega vectors
c   at 4 corners.
c
          OMEX(INM)=(OMEX(IN1)+OMEX(IN2)+OMEX(IN3)+OMEX(IN4))/4.0D0
          OMEY(INM)=(OMEY(IN1)+OMEY(IN2)+OMEY(IN3)+OMEY(IN4))/4.0D0
          OMEZ(INM)=(OMEZ(IN1)+OMEZ(IN2)+OMEZ(IN3)+OMEZ(IN4))/4.0D0
  170   CONTINUE
  180 CONTINUE
C
C  Mirroring vorticities on opposite 1/2 span of the wake.
C  NOTE: for symmetric wings only!
C
      DO 200 I=1,NWPTS
         NCOL=NRWAKE(KCIR,I)-1
        DO 190 J=1,NCOL/2
          IN1=NODE+(I-1)*(3*NCOL+2)+J
          IN2=NODE+(I-1)*(3*NCOL+2)+2*NCOL+1+J
          INM=NODE+(I-1)*(3*NCOL+2)+NCOL+1+J
          K=NCOL-J+1
          INO1=NODE+(I-1)*(3*NCOL+2)+1+K
          INO2=NODE+(I-1)*(3*NCOL+2)+2*NCOL+2+K
          INOM=NODE+(I-1)*(3*NCOL+2)+NCOL+1+K
          OMEX(INO1)=-OMEX(IN1)
          OMEY(INO1)=OMEY(IN1)
          OMEZ(INO1)=-OMEZ(IN1)
          OMEX(INO2)=-OMEX(IN2)
          OMEY(INO2)=OMEY(IN2)
          OMEZ(INO2)=-OMEZ(IN2)
          OMEX(INOM)=-OMEX(INM)
          OMEY(INOM)=OMEY(INM)
          OMEZ(INOM)=-OMEZ(INM)
  190   CONTINUE
  200 CONTINUE
```

```
c
c    If NCOL is odd, append missing center square vorticities.
c
       IF(MOD(NCOL,2) .EQ. 1) THEN
         INL1=NODE+(I-1)*(3*NCOL+2)+NCOL/2+1
         OMEX(INL1+1)=-OMEX(INL1)
         OMEY(INL1+1)=OMEY(INL1)
         OMEZ(INL1+1)=-OMEZ(INL1)
         INL2=NODE+(I-1)*(3*NCOL+2)+(5*NCOL)/2+2
         OMEX(INL2+1)=-OMEX(INL2)
         OMEY(INL2+1)=OMEY(INL2)
         OMEZ(INL2+1)=-OMEZ(INL2)
         IMID=NODE+(I-1)*(3*NCOL+2)+(3*NCOL)/2+2
         OMEX(IMID)=0.0D0
         OMEY(IMID)=0.0D0
         OMEZ(IMID)=0.0D0
       END IF
C****************************************
C  Append wake c factors to CFACT array
C****************************************
c
c   Fill with ones
c
       DO 220 I=NELE+1,NWELE
            DO 210 J=1,3
                 CFACT(I,J)=1.0D0
  210       CONTINUE
  220  CONTINUE
c
c    Fill C-factor array for wake
c
       DO 240 I=NELE+1,NWELE
         DO 230 J=1,3
c
c    Determine node number
c
           NN=N(I,J)
c
c    Find the z-component in element's local frame of the node's
c    global normal (n-sub-z).
c
           ANLZ=ANORX(NN)*DIRCOS(I,3,1)+ANORY(NN)*DIRCOS(I,3,2)
     .          +ANORZ(NN)*DIRCOS(I,3,3)
c
c    Convert global vorticity at node (CAP OMEGA) into element's
c    local frame (little omega).
c
           OMLX=OMEX(NN)*DIRCOS(I,1,1)+OMEY(NN)*DIRCOS(I,1,2)
     .                 +OMEZ(NN)*DIRCOS(I,1,3)
           OMLY=OMEX(NN)*DIRCOS(I,2,1)+OMEY(NN)*DIRCOS(I,2,2)
     .                 +OMEZ(NN)*DIRCOS(I,2,3)
           OMLZ=OMEX(NN)*DIRCOS(I,3,1)+OMEY(NN)*DIRCOS(I,3,2)
     .                 +OMEZ(NN)*DIRCOS(I,3,3)
```

```
c
c   The c-factor is calculated using Eq.(2.4-20) of Mracek's
c   dissertation.
c
            WSQ=OMLX*OMLX+OMLY*OMLY+OMLZ*OMLZ
            IF(WSQ .GT. 1.0D-10) THEN
              CFACT(I,J)=DSQRT(WSQ)/DSQRT(WSQ+(OMLZ*OMLZ)/(ANLZ*ANLZ))
            ENDIF
 230      CONTINUE
 240    CONTINUE

        RETURN
        END
```

Appendix C: Subroutine VELWK

Subroutine VELWK calculates the disturbance velocity induced at a point in the flow by the wake which has been modeled as rows of continuous vortex sheets. The routine calls the already existing routines, VELBND, for each triangular panel in the wake, and VELVF for the variable strength edge cores around each wake row.

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                   C
C                          ***************                          C
C                          *    VELWK    *                          C
C                          ***************                          C
C                                                                   C
C       THIS ROUTINE CALCULATES THE VELOCITY INDUCED AT A POINT     C
C       BY THE WAKE WHICH HAS BEEN MODELED AS ROWS OF CONTINUOUS    C
C       VORTEX SHEETS. THE METHOD TRANSFORMS GLOBAL VORTICITY AT    C
C       A NODE INTO LOCAL COORDINATES AND MULTIPLIES IT BY THE      C
C       APPROPRIATE VEL OUTPUT.                                     C
C                                                                   C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE VELWK
      IMPLICIT REAL*8(A-H,O-Z)
      PARAMETER (ME=1000,MN=1000,MEQ=200,MUN=200,MC=25,MCI=100)
      PARAMETER (MW=25,MWP=25)
      COMMON/ELDATA/ NELE,NODE,N(ME,3),XNODE(MN),YNODE(MN),ZNODE(MN)
      COMMON/EDGEDA/ NOPEN,NCLOSE,NTOTCR,NEDG(MC,MCI,2),NCIRC(MC)
      COMMON/CONVEC/ NCOVCR,NCONCR(MCI),NCON(MC,MCI,2),NCONPO(MCI)
      COMMON/CORNER/ AELE(ME),BELE(ME),CELE(ME),DIRCOS(ME,3,3)
      COMMON/VELINP/ XP,YP,ZP,IDENTE,NODE1,NODE2
      COMMON/VELVFO/ VXVF(3),VYVF(3),VZVF(3)
      COMMON/GNUMBE/ G1(MC,MCI),G2(MC,MCI),G3(MC,MCI)
      COMMON/GBYEDG/ G1W(MC,MCI),G2W(MC,MCI),G3W(MC,MCI)
      COMMON/ERRORS/ IERROR,KERROR,MERROR,NERPOR
      COMMON/VELOUT/ VX(6),VY(6),VZ(6)
      COMMON/OMEGAL/ OMLOCX(3),OMLOCY(3)
      COMMON/GNORMS/ ANORX(MN),ANORY(MN),ANORZ(MN),CFACT(ME,3)
      COMMON/CONSTS/ FOURPI,PI,CUTOFF,CUTWK
      COMMON/GOMEGB/ OMEX(MN),OMEY(MN),OMEZ(MN)
      COMMON/VCFINP/ X1,Y1,Z1,X2,Y2,Z2,VXCF,VYCF,VZCF
      COMMON/STRWAK/ GAVE(MC,MCI),GWAKE(MC,MCI,MW),NRWAKE(MC,MW)
      COMMON/SPLITW/ ISPLIT(MC,MCI,MW),DIMINI,ISPER(MCI,MW)
      COMMON/TIMESD/ TIME,DTIME,NMOTIO,NCURTM,NTIME,NSTIME,NWPTS,MAXWK
      COMMON/VBOUND/ VBX,VBY,VBZ
      COMMON/VEWAKE/ VWKX,VWKY,VWKZ
      COMMON/CNTERS/ KCIR,NWELE
```

```
C
C   INITIALIZING THE VELOCITIES
C
      VWKX=0.0D0
      VWKY=0.0D0
      VWKZ=0.0D0
C
C   LOOPING THROUGH THE WAKE ELEMENTS FOR EACH ONES CONTRIBUTION
C
      DO 10 I=NELE+1,NWELE
           IDENTE=I
C
C   FINDING THE LOCAL VORTICITY AT EACH NODE OF ELEMENT I
C
           DO 20 J=1,3
                ANX=ANORX(N(I,J))*DIRCOS(I,1,1)
     .             +ANORY(N(I,J))*DIRCOS(I,1,2)
     .             +ANORZ(N(I,J))*DIRCOS(I,1,3)
                ANY=ANORX(N(I,J))*DIRCOS(I,2,1)
     .             +ANORY(N(I,J))*DIRCOS(I,2,2)
     .             +ANORZ(N(I,J))*DIRCOS(I,2,3)
                ANZ=ANORX(N(I,J))*DIRCOS(I,3,1)
     .             +ANORY(N(I,J))*DIRCOS(I,3,2)
     .             +ANORZ(N(I,J))*DIRCOS(I,3,3)
                A11=CFACT(I,J)*(ANX*ANX+ANZ*ANZ)/(ANZ*ANZ)
                A12=CFACT(I,J)*ANX*ANY/(ANZ*ANZ)
                A22=CFACT(I,J)*(ANY*ANY+ANZ*ANZ)/(ANZ*ANZ)
                WOMEGX=OMEX(N(I,J))*DIRCOS(I,1,1)
     .                +OMEY(N(I,J))*DIRCOS(I,1,2)
     .                +OMEZ(N(I,J))*DIRCOS(I,1,3)
                WOMEGY=OMEX(N(I,J))*DIRCOS(I,2,1)
     .                +OMEY(N(I,J))*DIRCOS(I,2,2)
     .                +OMEZ(N(I,J))*DIRCOS(I,2,3)
                OMLOCX(J)=A11*WOMEGX+A12*WOMEGY
                OMLOCY(J)=A12*WOMEGX+A22*WOMEGY
   20      CONTINUE
C
C   CALLING THE VELOCITY FOR ELEMENT I
C
           CALL VELE
C
C   ADDING TO THE OTHER ELEMENTS
C
           DO 30 J=1,3
                VWKX=VWKX+VX(J)*OMLOCX(J)
     .                 +VX(J+3)*OMLOCY(J)
                VWKY=VWKY+VY(J)*OMLOCX(J)
     .                 +VY(J+3)*OMLOCY(J)
                VWKZ=VWKZ+VZ(J)*OMLOCX(J)
     .                 +VZ(J+3)*OMLOCY(J)
   30      CONTINUE

   10 CONTINUE
```

```
C
C   ADDING THE INFLUENCE OF THE CORES ALONG THE EDGES OF
C   EACH WAKE ROW.
C
      DO 50 I=NTOTCR+1,NTOTCR+NWPTS
           DO 40 J=1,NCIRC(I)
c             IF(J .GT. NCIRC(I)/2-1 .AND. J .LT. NCIRC(I)-1) THEN
c               NODE1=NEDG(I,J,2)
c               NODE2=NEDG(I,J,1)
c             ELSE
                NODE1=NEDG(I,J,1)
                NODE2=NEDG(I,J,2)
c             END IF
                     CALL VELVF
                     VWKX=VWKX+G1W(I,J)*VXVF(1)
     .                      +G2W(I,J)*VXVF(2)
     .                      +G3W(I,J)*VXVF(3)
                     VWKY=VWKY+G1W(I,J)*VYVF(1)
     .                      +G2W(I,J)*VYVF(2)
     .                      +G3W(I,J)*VYVF(3)
                     VWKZ=VWKZ+G1W(I,J)*VZVF(1)
     .                      +G2W(I,J)*VZVF(2)
     .                      +G3W(I,J)*VZVF(3)
   40      CONTINUE
   50 CONTINUE

      RETURN
      END
```

Bibliography

1. Mracek, Curtis P. Unsteady Potential-Flow Solution Using Vortex Panels Coupled With Dynamics and Controls. PhD dissertation. Virginia Polytechnic Institute and State University, Blacksburg, VA, July, 1988.

2. Karamcheti, Krishnamurty. Principles of Ideal-Fluid Aerodynamics. Malabar,Florida: Robert E. Krieger Publishing Co., Inc., 1966.

3. Beran, Philip. Subroutine "MSOLV." Fortran subroutine to solve a system of linear equations. Air Force Institute of Technology, Wright Patterson AFB, Ohio.

Vita

Major Robert J. Papka [redacted]

[redacted] He graduated from Billings Senior High School in 1974 and entered the U.S. Air Force Academy three weeks later. He received a Bachelor of Science in Aeronautical Engineering and his commission in May 1978. He completed pilot training in June 1979 at Vance AFB, OK, and remained for a three year assignment as a T-37 instructor pilot. He has since flown as a C-140 Special Air Missions Aircraft Commander at Ramstein AB, W. Germany, and was a C-141 aircraft commander at McChord AFB, WA until entering the School of Engineering, Air Force Institute of Technology, in May 1988.

[redacted]

[redacted]

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; Distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GAE/ENY/89D-26	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION School of Engineering; 6b. OFFICE SYMBOL (If applicable) AFIT/ENY	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State, and ZIP Code) Air Force Institute of Technology Wright-Patterson AFB OH 45433-6583	7b. ADDRESS (City, State, and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION; 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
MODELING THE WAKE AS A CONTINUOUS VORTEX SHEET IN A POTENTIAL-FLOW SOLUTION USING VORTEX PANELS

12. PERSONAL AUTHOR(S)
Robert J. Papka, B.S., Major, USAF

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
MS Thesis	FROM ____ TO ____	1989 December	94

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Aerodynamic characteristics, Vorticies, Vortex Shedding, Trailing Vorticies, Fluid Dynamics
20	04		

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Thesis Advisor: Curtis P. Mracek, Capt, USAF
Assisitant Professor of Aerospace Engineering
Department of Aeronautics and Astronautics

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT [✓ UNCLASSIFIED/UNLIMITED] ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL Capt Curtis P. Mracek, Asst. Professor	22b. TELEPHONE (Include Area Code) 513-255-2362; 22c. OFFICE SYMBOL ENY

UNCLASSIFIED

An investigation was made to determine the advantages of modeling the wake as a continuous vortex sheet in an unsteady potential-flow computer model. The existing program modeled the body as a continuous vortex sheet, and modeled the wake as discrete vortex cores employing principles of vortex lattice methods. A method is developed to approximate the wake with triangular vortex panels. The method accounts for redistributing vortex strength as the panels expand and deform. Comparisons are made with results of the original program for a rectangular wing at varying angles of attack. Limited results suggest the method is valid, and areas for further study are recommended.